LAUREA
AMMATTIKORKEAKOULU
Yhdessä enemmän

Kuvapankkipalvelun kehitys BTJ Finland Oy:lle

Karjalainen, Eemeli

2016 Laurea

Laurea-ammattikorkeakoulu

Kuvapankkipalvelun kehitys BTJ Finland Oy:lle

Eemeli Karjalainen
Tietojenkäsittelyn koulutusohjelma
Opinnäytetyö
Huhtikuu, 2016

Laurea-ammattikorkeakoulu **Tiivistelmä**
Tietojenkäsittelyn koulutusohjelma
Tradenomi (AMK)

Eemeli Karjalainen

Kuvapankkipalvelun kehitys BTJ Finland Oy:lle

Vuosi 2016 Sivumäärä 41

Tässä opinnäytetyössä kerrotaan kuvapankkiprojektista, joka toteutettiin osana työharjoittelua yrityksessä BTJ Finland. Työssä kuvaillaan myös projektin aikana käytettyä jatkuvan kehittämisen menetelmää, eli CI-mallia. Työn tavoitteena oli projektin kuvaamisen ohella käsitellä jatkuvan kehittämisen menetelmän hyötyjä, haittoja sekä siihen sisältyviä haasteita. Projektin tavoitteena puolestaan oli kehittää kuvapankkipalvelu, joka tulisi ensisijaisesti Helsingin kaupunginkirjaston käyttöön.

Työssä esitellään ensin tutkimukseen liittyvät lähtökohdat ja tavoitteet. Tämän jälkeen esitellään itse projekti pääpiirteittäin. Projektin esittelyä seuraa CI-mallin, sekä muutaman tämän lukuisista sovelluksista, esittely. Myöhemmin työssä kuvaillaan tarkemmin projektin vaiheita, sekä sitä kuinka CI-mallin soveltamista käytännössä toteutettiin. Lopuksi johtopäätöksissä kootaan keskeiset havainnot kehitystyöhön liittyen, sekä tämän jälkeen päällimmäiset ajatukset ja kokemukset projektiin liittyen, kappaleessa oman oppimisen arviointi.

CI-mallin käyttäminen havaittiin pääasiassa toimivaksi ratkaisuksi projektin aikana. Tähän vaikutti varsinkin ohjelmistokehitystyön luonne, jolle oli keskeistä jatkuvan testaamisen ja tarpeiden kartoituksen ohella etenkin asiakaslähtöisyys. Ohjelmistokehityksessä CI-mallin käyttäminen rinnastuu usein myös ns. ketterien menetelmien käyttöön, jossa tärkeänä piirteenä voidaan pitää etenkin joustavuutta ja suoraa viestintää.

Asiasanat: Asiakaslähtöisyys, ohjelmistokehitys, PHP, rajapinta

Laurea University of Applied Sciences **Abstract**
Degree Programme in Business Information Technology
Bachelor's Thesis

Eemeli Karjalainen

Development of an image database service for BTJ Finland Oy

Year 2016 Pages 41

This Bachelor's thesis describes the image database project that the author participated in during an internship in BTJ Finland, and a continual improvement process (CI) that was implemented during the project. The aim of the thesis was to review possible benefits and disadvantages of CI, alongside documenting the project. The primary goal of the project itself was to develop image database service primarily for the use of Helsinki City Library.

The thesis introduces foremost the starting points and goals of the study, after which the project is described briefly. Afterwards CI is introduced with some its applications, followed by the project and its steps which are discussed in detail. The implementation of CI during the project is also examined.

The usage of CI proved to be workable solution for the most part of the project. In addition to continuous testing and mapping of the needs, customer orientation also proved to be an essential part.

Keywords: API, Customer orientation, PHP, Software development

Käytetyt erikoistermit ja lyhenteet

API	Application Programming Interface, ohjelmointirajapinta
CI-malli	Continual improvement, jatkuvan kehittämisen malli
CSS	Cascading Styling Sheets, www-dokumenteille kehitetty tyylioh-jeiden laji
DMAIC-menetelmä	Define (määrittely), Measure (mittaus), Analyse (analysointi), Improve (parannus), Control (ohjaus)
FTP	File Transfer Protocol, kahden tietokoneen välinen tiedostonsiir-tojärjestelmä
Hash	Tiiviste eli hajautusarvo, jolla tieto saadaan pienempään tilaan tiivistealgoritmin avulla (esim. MD5)
HTML	Hypertext Markup Language
IP-osoite	Verkkosovittimien yksilöimiseen käytettävä numerosarja
Javascript	Dynaamisesti tyypitetty, tulkattava oliopohjainen komentosarja-kieli
MySQL	Yleisin relaatiotietokantatyyppi, joka käyttää SQL-kieltä
PHP	PHP: Hypertext Preprocessor
SQL	Structured Query Language, tietokantajärjestelmien kyselykieli
Validointi	Validointi, eli vahvistaminen on prosessi, jossa tarkistetaan, että prosessin kohde täyttää jotkin tietyt kriteerit
XML	Extensible Markup Language, rakenteellinen kuvauskieli isojen tietomassojen jäsentelyyn

Sisällys

1 Johdanto

Suorittaessani työharjoittelua BTJ Finland Oy:llä osallistuin Armas-kuvapankkipalvelun kehittämiseen. Projektin tarkoituksena oli ensisijaisesti kehittää Helsingin kaupunginkirjaston käyttöön palvelu, jonka kautta asiakkaalla olisi pääsy haluamiensa tuotteiden medioihin ja näihin liittyviin tuotetietoihin. Medialla tarkoitettiin tässä tapauksessa ensisijaisesti kuvatiedostoja, mutta tulevaisuudessa näiden rinnalle saataisiin myös muuta mediaa, kuten ääntä ja liikkuvaa kuvaa.

Minulle oli kerrottu, että projektista olisi mahdollista saada harjoittelupaikan lisäksi myös aihe opinnäytetyötä varten. Olin luonnollisesti alusta asti kiinnostunut sekä tilaisuudesta hankkia työkokemusta projektityöskentelystä ja ohjelmistokehityksestä että itse projektin aiheesta.

Tässä opinnäytetyössä on tarkoituksena kuvailla Armas -kuvapankkipalvelua jonka kehityksessä olin mukana kesäkuusta lokakuuhun 2012, sekä sen aikana käytettyä jatkuvan kehityksen mallia. Kuvapankkiprojektin ensimmäinen varsinainen tavoite oli kyselyrajapinnan kehittäminen tuotantoa varten. Tämän lisäksi projektiin kuului esimerkiksi tuotetietokannan sekä kuvapankin graafisen käyttöliittymän suunnittelu ja toteutus. Itse järjestelmä ja kyselyrajapinta kehitettiin alun perin Helsingin kaupunginkirjaston käyttöön, mutta projektin aikana puhuttiin myös mahdollisuudesta myydä sitä tulevaisuudessa muillekin asiakkaille.

Tällä hetkellä kuvapankin kyselyrajapintaa voi kokeilla esimerkiksi hakemalla ensin omaa suosikkiteostaan osoitteessa http://www.helmet.fi, menemällä tuotteen omalle sivulle ja klikkaamalla vasemmalla sijaitsevaa tuotteen kansikuvaa. Tämä aktivoi toiminnon, jolla kyselyrajapinta avaa tuotetietosivun. Tähän näkymään kyselyrajapinta tulostaa valitun tuotteen kuvan mikäli saatavilla, perustietoja sekä esittelytekstin.

Armas-kuvapankkipalvelu on tietokanta, joka sisältää kuvia BTJ:n tuotteista sekä näihin liittyviä tietoja. Asiakkaalla on lähtökohtaisesti pääsy sekä kuvatiedostoihin että näiden tuotetietoihin.

Tulevissa luvuissa esitellään ensin kyseisen jatkuvan kehittämisen (Continual Improvement Process, CI) mallin pääperiaatteet ja esimerkkejä sen sovelluksista. Tämän jälkeen on tarkoitus käsitellä tarkemmin itse projektin sisältö teknisiä määrittelyjä myöten, mutta kaiken kaikkiaan paino pidetään jatkuvan kehityksen mallin näkökulmassa ja sen soveltamisessa käytäntöön. Järjestelmän ja CI-mallin ohella käyn läpi omat kokemukseni sekä päällimmäisinä olleet ajatukset itse projektiin osallistumisesta.

Tarkemmin ottaen pyrinkin erittelemään jatkuvan kehittämisen mallin eri ominaisuuksia haitoista hyötyihin, ja kuvailemaan tässä tapauksessa kuvapankkiprojektia CI-mallin näkökulmasta sekä sitä, kuinka mallin soveltaminen käytännössä toimi. Lopuksi pyrin vielä johtopäätöksissä kiteyttämään muutamia pääajatuksia ja havaintoja joita olen tässä työssä tuonut esille sekä vastaamaan mahdollisesti muutamiin keskeisiin kysymyksiin aiheeseen liittyen.

2 Työn lähtökohdat ja tavoitteet

Tutkimuksessani vastaan seuraavanlaisiin kysymyksiin:

- Mitä hyötyä on jatkuvan kehittämisen mallista nykyaikaisessa ohjelmistokehityksessä?
- Miksi ketterä ohjelmistokehitysmalli on hyvä?
- Mitä haasteita ja ongelmia mallin käyttämiseen liittyy?

Niin sanottujen ketterien menetelmien käyttö alkaa nykyaikaisessa projektityöskentelyssä olla jo melkein itsestäänselvyys. Monille ei välttämättä tulisi enää edes mieleen kyseenalaistaa ketterien menetelmien käyttöä. On kuitenkin edelleen myös heitä, jotka eivät välttämättä tunnista ketterän menetelmän välittömiä hyötyjä. He saattavat kuvitella, että sovelluskehityksessä voidaan noudattaa vanhanaikaisempaa vesiputousmallia muistuttavaa menetelmää, jossa tuote jalostetaan kerralla täydelliseksi pitkän laadunvalvontaprosessin lopputuloksena.

Vesiputousmallista on luonnollisesti olemassa myös monta erilaista muunnelmaa, mutta lähes jokaisessa näistä on erotettavissa tietyt samat vaiheet. Näitä vaiheita on yhteensä kolme; määrittely-, suunnittelu ja toteutusvaihe. Määrittelyvaihetta edeltää usein esitutkimukseksi (feasibility study, preliminary analysis) tai tarvekartoitukseksi (requirements study) kutsuttu vaihe (Haikala & Märijärvi 2004, 37). Tällainen lähestymistapa saattoi soveltua hyvin esimerkiksi pitkissä sarjoissa valmistettaviin teollisuustuotteisiin, mahdollisesti myös ohjelmistotuotantoon, mutta ohjelmistokehityksessä siitä ei välttämättä ainakaan sellaisenaan ole hyötyä. Asiakkaan tarpeita ja projektin kustannuksia voidaan toki kartoittaa ennen ohjelmistoprojektin aloittamista, mutta kartoittaminen itsessään ei kovinkaan usein riitä lopullisen tuotteen suunnitelmaksi. Lisäksi kattava tarvekartoitus saattaa syödä turhankin paljon resursseja tuottamaansa hyötyyn nähden. Varsinkin kun ottaa huomioon, että projektin tavoitteet ja asiakkaan tarpeet saattavat muuttua hankkeen edetessä, ei turhan kattavaan kartoitukseen välttämättä aina kannata tuhlata resursseja.

Usein jopa yhden ja saman ohjelmiston käyttäjillä voi olla keskenään poikkeavia vaatimuksia ja tarpeita ominaisuuksien suhteen. Keskeistä tässä tapauksessa onkin usein suora yhteys asiakkaaseen ja siihen kuinka asiakas käyttää tuotetta eli sovellusta. Ketteriä menetelmiä käyttävä tiimi kehittää ohjelmaa ominaisuus kerrallaan ja näin ei tuhlata resursseja turhien toimintojen kehittämiseen. Halutaan vastata nopeasti muutoksiin. Kaiken perusta on toimiva kehitysympäristö, julkaisuprosessin tunteminen ja nopea julkaisusykli (Väyrynen 2009, 45).

Tässä työssä kuvaan järjestelmän kehitysprojektin eri vaiheita. Järjestelmän suunnittelu ja toteutus tehtiin yhteistyönä, ja järjestelmästä käytettiin projektin aikana nimitystä Armas-kuvapankkipalvelu, vaihtoehtoisesti myös pelkkä Armas tai kuvapankki. Projektin aikana käytettyjä kehitysmenetelmiä olivat mm. viikkopalaverit, joissa käytiin tavallisesti läpi edellisen viikon aikaansaannokset ja tulevan viikon tavoitteet, sekä erilaiset tilaisuudet määrittelyjä ja ohjeistuksia varten. Vastaaviin menetelmiin voidaan lukea myös yhteydenpito asiakkaaseen. Esimieheni oli pääasiassa vastuussa yhteydenpidosta, ja siitä että asiakkaan antama palaute tuli koko kehitystiimin tietoon. Kuvaan projektissa käytettyjä kehitysmenetelmiä, sekä kerron myös muutamassa kohtaa hieman tarkemmin sovelluskehityksestä ja siitä kuinka toteutin eri toiminnallisuuksia ohjelmointitasolla. Käytetyistä erikoistermeistä ja käsitteistä huolimatta en kuitenkaan keskity varsinaisesti ohjelmointiin, käytettyihin ohjelmointikieliin enkä etenkään näiden ominaisuuksien vertailuun.

Keskittyminen jatkuvan kehittämisen malliin ei ollut kovin helppo valinta, kun ottaa huomioon, että lopputyö olisi ollut mahdollista toteuttaa myös pelkästään itse projektin näkökulmasta. Tässä lähestymistavassa toisin sanoen vain kuvailtaisiin koko projekti kaikkine vaiheineen suunnittelusta ensimmäiseen tuotantoversioon saakka. Ensimmäisessä vaihtoehdossa aineistona voi käyttää melkeinpä pelkästään projektin dokumentaatioita, testituloksia sekä ylipäätään lähes kaikkia materiaaleja taulukoista kaavioihin. Pelkkään projektin kuvaamiseen keskittyessä muodostuukin helposti ongelmia, kun lähdemateriaalin niukkuus ei salli kovinkaan laajamittaista asioiden käsittelyä. Yhtenä ratkaisuna tähän olisi tietysti voinut olla CI-mallin ottaminen näkökulmaksi, mutta niin, että työn painopiste pysyisi itse projektissa. Päätin ettei kyseinen menettely olisi kuitenkaan sopiva, saati sitten tuottaisi kiinnostavaa lopputulosta, vaan halusin toteuttaa työni juuri päinvastaisella tavalla.

Niinpä otin jatkuvan kehittämisen mallin varsinaiseksi pääaiheeksi, sillä siten saisin mahdollisesti mielekkäämmän lopputuloksen joutuessani soveltamaan ensimmäistä vaihtoehtoa enemmän ulkopuolisia lähteitä. Voin tarvittaessa hyödyntää myös projektiaineistoa, jotta näkökulma pysyy käytännönläheisenä tilanteen ja aihepiirin niin vaatiessa. Näin käyttämäni aineisto on jo lähtökohtaisesti laajempi, ja lopputulos lukijan kannalta mielenkiintoisempi.

3 Projektin esittely

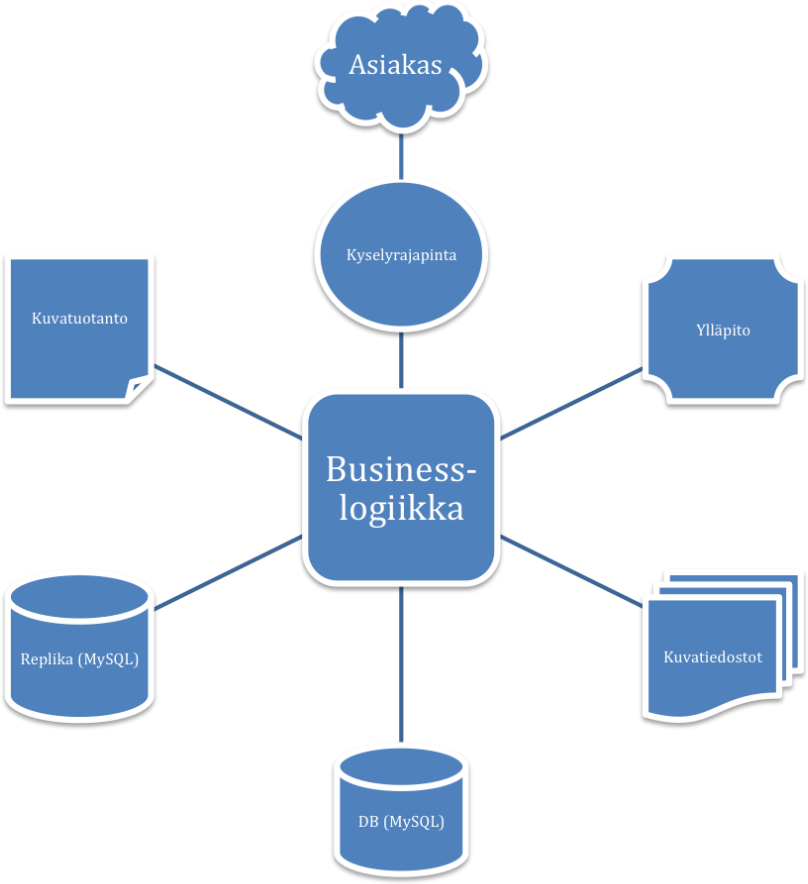
Timo Lehtimäki (2006, 5-8) on kertonut projekteista, niiden eri vaiheista sekä siitä, miksi juuri projekti on hyvä kehittämisen muoto. Projektille määritellään tarkoin sen tavoite, aikataulu sekä tavallisesti myös kustannusarvio. Kun sovitut tavoitteet ollaan saavutettu, ”vastuunsa täytettyään se (projekti) loppuu.(…) Resurssit sitoutuvat vain siksi aikaa, kun niitä tarvitaan. Kun projekti on ohi, ihmiset voivat siirtyä seuraaviin töihin.” (Lehtimäki 2006, 5.) Vastaavasti pelkästään projektia varten palkatut työntekijät päättävät määräaikaisen työsuhteensa, kun projekti on ohi. Näin resursseja pyritään hyödyntämään mahdollisimman tehokkaasti. Omalla kohdallani työharjoittelu kesti viisi kuukautta, kuten olimme ennalta sopineet.

BTJ Finland perustettiin alun perin vuonna 1961 nimellä Kirjastopalvelu, mutta vuodesta 1995 eteenpäin yritys on ollut osa ruotsalaista BTJ Group –konsernia. Tänä päivänä BTJ on ruotsalaisten lisäksi Suomen kirjastoseuran omistuksessa. Suomen kirjastoseura omistaa yrityksestä 20 prosenttia.

BTJ välittää ja maahantuo erilaista aineistoa pääasiassa kirjastoille, mutta myös esimerkiksi kouluille, näiden omille kirjastoille, yrityksille sekä julkishallinnoille. Yleisten kirjastojen aineistonvälityksessä BTJ onkin Suomessa markkinajohtaja.

BTJ:n toimisto sijaitsee Lauttasaaressa. lyhennys BTJ tulee sanoista Bibliotekstjänst. BTJ:n arvot ovat asiakaslähtöisyys, yhteistyö, liiketoiminnallisuus, reiluus ja luovuus. Näitä arvoja pyrittiin luonnollisesti noudattamaan myös projektin aikana. Etenkin asiakaslähtöisyys oli varsin keskeisessä osassa kuvapankkipalvelua kehittäessä.

Projektin tarkoituksena oli kehittää kuvanpankkipalvelu, joka helpottaisi BTJ:n eri tuotteiden, näihin liittyvien medioiden sekä tuotetietojen käsittelyä. Palvelua käyttävät niin BTJ:n asiakkaat, työntekijät, kuin ylläpito joka tekee siihen tarvittavat päivitykset ja muutokset, liittyen esimerkiksi käyttöoikeuksiin. Kuvio 1. auttaa hahmottamaan järjestelmän rakennetta, jossa eri osaset muodostavat yhdessä toimivan kokonaisuuden eli business-logiikan. Tuotteiden tietoja ja metadataa varten luotiin oma MySQL-tietokanta. MySQL on relaatiotietokantoja käsittelevä ilmainen tietokannanhallintajärjestelmä. Relaatiotietokannoissa tieto on järjestetty tauluihin, joiden on mahdollista viitata toisiinsa keskinäisten suhteidensa avulla. Eri tietokantatyypeistä relaatiotietokantoja käytetään tällä hetkellä ylivoimaisesti eniten. (Heinisuo 2007, 38.)



Kuvio 1: Business-logiikka

Termillä rajapinta tarkoitetaan määritelmää, jonka mukaan eri ohjelmat voivat tehdä pyyntöjä ja vaihtaa tietoja eli keskustella keskenään. Ohjelmointirajapinta (API) määrittää sen, miten ohjelmisto tarjoaa tietoja tai palveluita sovelluksille tai muille tietojärjestelmille. Rajapinta voi käyttötarkoituksesta riippuen olla joko avoin tai suljettu. Ollakseen avoin on rajapinnan täytettävä tietyt ehdot. Näitä ehtoja ovat dokumentointi, eli rajapinnan määrittelyjen sekä järjestelmän tietojen on oltava avoimesti saatavilla verkon kautta, käyttöönotettavuus, eli mahdollisuus rajapinnan käyttöönottoon ilman ylläpitäjän tai järjestelmätoimittajan toimia sekä testattavuus. Testattavuudella tarkoitetaan joko avointa pääsyä itse tuotantojärjestelmään, tätä simuloivaan testijärjestelmään joka sisältää realistista testidataa tai vaihtoehtoisesti sitä, että järjestelmä on vapaasti ladattavissa käyttäjän omaan käyttöön. (Kivekäs 2014.)

Tässä tapauksessa kyseessä oli siis suljettu rajapinta ja sovellus, jolla tietokanta, itse kuvatiedostot ja käyttäjät kommunikoivat keskenään. Termi kysely puolestaan viittaa relaatiotietokantojen kysely- ja määrittelykieli SQL:ään (Structured Query Language).

Rajapinnan toteutuskieleksi sovittiin käytettäväksi PHP, sillä monien sen ominaisuuksista katsottiin sopivan varsin hyvin yhteen sovittujen tavoitteiden kanssa. Sen lisäksi PHP:n avulla on mahdollista käsitellä helposti MySQL-tietokannassa olevaa tietoa. PHP mahdollistaa tämän tarjoamalla komentoja ja rakenteita, joilla voi välittää SQL-komentoja tietokantaan. Näin ollen on mahdollista paitsi lukea tietokannan taulujen sisältämää tietoa siten, että tämä tulostuu tavalliseksi www-sivuksi tai osaksi sitä, voidaan sen lisäksi tallentaa HTML-lomakkeille syötettyä tietoa. PHP-koodia on monen muun ohjelmointikielen tapaan mahdollista sijoittaa HTML-muotoilujen väliin, ja tästä ominaisuudesta olikin jonkin verran hyötyä sekä kyselyrajapinnan että tuotantoprosessin tuen toteutuksessa. Kolmas olennainen syy PHP:n valintaan ohjelmointikieleksi oli, että hallitsin tämän toistaiseksi hyvin.

Kyselyrajapinnan kautta saamallaan kuvalinkillä asiakas voi upottaa kuvan sivustolleen lisäämällä lähdekoodiinsa linkin kyselyrajapintaan. Alun perin oli tarkoituksena, että toteuttaisin kuvapankin ylläpitotyökalun käyttämällä C#-ohjelmointikieltä. Toteutuskieleksi valittiin kuitenkin kyselyrajapinnan tapaan PHP. Ylläpitotyökalun avulla kuka tahansa työntekijä voisi graafista käyttöliittymää hyödyntämällä lisätä uusia tuotteita, kuvia tai tietoja sekä lisätä ja poistaa asiakas-, sekä käyttäjätilejä. Tietojen ohella myös asiakkaiden käyttöoikeuksia on mahdollista muuttaa.

Tuotetietojen ylläpitoa varten BTJ Finlandin käytössä oli jo aikaisemmin olemassa tietokanta, jota päivitetään jatkuvasti sitä mukaan, kun Ruotsissa sijaitsevalle palvelimelle lisätään uusia tuotteita sekä näiden tietoja.

Projektimallina käytettiin CI-menetelmää, eli jatkuvan kehittämisen mallia. Asiakaslähtöisyyden ohella tämä näkyi käytännössä lähinnä työskentelytapojen joustavuudessa, ohjelmiston jatkuvana testaamisena sekä uusien lisäominaisuuksien kehittämisenä palautteiden ja testitulosten pohjalta.

Taulukossa 1 on esitettynä alkuperäinen suunnitelma projektin etenemisaikataulusta suunnittelusta tuotantoon siirtämiseen asti. Järjestelmän kehitystä jatkettiin vielä senkin jälkeen kun kyselyrajapinta oli siirretty tuotantoon, ja tästä kerrotaan tarkemmin luvussa 6. Pääasiassa kullekin viikolle oli varattuna aina jonkin tietyn ominaisuuden toteuttaminen. Vaiheiden etenemistä seurattiin ja käytiin läpi projektipalavereissa sekä maantaisin pidettävissä viikkopalavereissa. Seuraavassa avaan enemmän näitä tehtävävaiheita.

Viikko	Tehtävä
26	Speksit ja kehitysympäristö
27	Palvelinohjelmiston runko Tietokannan ensimmäinen versio Tiedostorakenne
28	Kyselyrajapinnan ensimmäinen versio Replika-integraation aloitus
29	Autentikointi Kuvien ryhmittely ja oikeudet
30	Tilastointi
31	Asiakaskohtainen parametrointi
32	Tuotantoon siirto ja testaus
33	Go live

Taulukko 1: Projektin viikkoaikataulu

Viikolla 26 teimme speksit eli projektin määrittelyn ja laitoimme kehitysympäristön valmiiksi. Kehitysympäristönä toimi palvelin johon oli pääsy kaikilla osaston jäsenillä. Tavallisesti ohjelmisto ladattiin muokkausta varten palvelimelta omalle koneelle FTP-yhteyden avulla. Tämän jälkeen ohjelmistokoodin uusi versio päivitettiin palvelimelle.

Seuraavaksi viikolla 27 teimme rungon palvelinohjelmistoa varten, tietokannan ensimmäisen version sekä tiedostorakenteen. Tietokannan ensimmäisessä versiossa ei vielä ollut varsinaista sisältöä eli tuotteita ja näiden tietoja, joten tein aluksi manuaalisesti syöttämällä testidataa ohjelmiston testaamista varten. Tiedostorakenne piti luonnollisesti toteuttaa niin että kaikki tiedostot arkistoitaisiin järkevästi ja loogisesti. Viimeistään tuotantovaiheessa kuvatiedostoja tulisi palvelimelle sen verran paljon, että kaikkien tiedostojen pitäminen yhdessä hakemistossa todennäköisesti kuormittaisi ja hidastaisi palvelimen toimintaa kohtuuttomasti.

Sitten viikolla 28 tein valmiiksi ensimmäisen version kyselyrajapinnasta. Aloitimme myös replika-integraation, eli kuvapankin varsinaisen sisällön siirtämisen replika-palvelimelta kehitysympäristöön. Replika varastoi kaikki Ruotsista tulevat tuotetiedot ja mediat. Rajapinta on ensisijaisesti tietokantahakuja varten. Tarkoituksena oli alun perin että ylläpitoa varten luo-

taisiin oma rajapinta joka mahdollistaisi materiaalin lisäämisen ja toimintojen hallinnoinnin. Kuvapankkiin vaadittiin myös erillinen integraatio uusien tuotetietojen päivittämistä varten replika-palvelimelta. Myöhemmin tuotetiedot päivittyisivät automaattisesti sitä mukaan kun uutta replikalle tulisi uutta sisältöä.

Viikko 29 oli varattu ensisijaisesti autentikointiin, eli käyttöoikeuksien hallintaan sekä kuvien ryhmittelyyn. Kuvat ryhmitellään ensisijaisesti metadatan perusteella. Tämän tarkoituksena oli helpottaa hakujen tekemistä rajaamalla hakutuloksia tunnisteiden mukaan. Kuvien oikeuksilla määritellää, ketkä kaikki voivat nähdä kuvat. Kuvien käyttöoikeuksia on mahdollista rajoittaa jonkin tietyn asiakkaan ulottumattomiin tai vaihtoehtoisesti kuviin itseensä liitetyillä rajoituksilla.

Seuraavana oli tarkoitus aloittaa tilastointi. Tämän toiminnon oli määrä koota tietoa kuvapankkipalvelun käytöstä seurantaa varten. Tilastoa kootaan paitsi olemassa olevien kuvatiedostojen määrästä ja tuotannosta, niin myös kuvapankin käytöstä. Toiminnon avulla on esimerkiksi mahdollista seurata kyselyrajapinnalla suoritettujen hakujen ja kyselyjen määrää.

Sen jälkeen aloitimme asiakaskohtaisen parametroinnin toteuttamiseen viikolla 31.
Kuten aikaisemmin on todettu, on asiakkailla usein keskenään erilaisia käyttötarpeita palvelun toiminnallisuuksien ja ominaisuuksien suhteen. Asiakaskohtaisen parametroinnin ideana oli tarjota mahdollisuus muokata palvelussa olevia ominaisuuksia halutunlaisiksi niin että nämä muutokset jäävät voimaan pysyvästi. Näin palvelu kykenisi tunnistamaan asiakkaan ja muistamaan tätä varten tehdyt muutokset joka kerta kun palvelua käytetään.

Viimein viikolla 32 oli tarkoitus siirtää ohjelmisto kehityspalvelimelta tuotantoon sekä aloittaa testaaminen. Tämän jakson aikana saimme jo joitain ehdotuksia lisäominaisuuksista. Ensisijaisesti tarkoituksena oli varmistaa, että tuotantopalvelin toimisi moitteettomasti, jotta ohjelmisto voitiin saattaa seuraavalla viikolla viimein tuotantokäyttöön.

4 CI-mallin esittely

Continual Improvement Process, eli CI-malli on menetelmä, jossa pyritään parantamaan jatkuvasti jonkin palvelun ominaisuuksia, kuten tehokkuutta, hyötysuhdetta ja joustavuutta. Jatkuvassa kehittämisessä pyritään joko parantamaan olemassa olevaa palvelua vaiheittain ajan myötä tai tekemään ns. läpimurto lyhyen ajan sisällä. Jatkuvan kehittämisen apuna käytetään asiakkaiden antamaa palautetta sekä tietoa, jota kerätään koko prosessin ajan.

CI-mallista on olemassa useita eri sovelluksia, joista jäljempänä ovat käsiteltyinä Kaizen, Lean ja Six Sigma. Jokainen näistä sisältää jossakin määrin saman pääperiaatteen tuotteen ja palvelun jatkuvasta kehityksestä, mutta yleensä kuitenkin omilla lisäsäännöillä.

Yleisesti ottaen CI–mallin hyvänä puolena voidaan pitää sen mahdollisuutta saattaa projekti yllättävänkin nopeasti tuotantovaiheeseen, ja näin päästä testaamaan tuotteen tai palvelun toimivuutta asiakkaiden käytössä lähes välittömästi. Asiakkaan antaman palautteen perusteella pystytään laatimaan parannusehdotuksia. Kehitettävän tuotteen tai palvelun ensimmäisen version ei ole tarkoituskaan olla vielä täydellinen, vaan sen on tarkoitus antaa asiakkaalle mahdollisuus kokeilla eri ominaisuuksia ja arvioida näiden sopivuutta tämän omiin tarpeisiin.

4.1 Kaizen

Kaizen on japania ja tarkoittaa parannusta tai muutosta parempaan (kai = muutos, zen = hyvä).

Kaizen otettiin alunperin käyttöön Japanissa toisen maailmansodan jälkeen pääasiassa amerikkalaisten asiantuntijoiden myötävaikutuksella. Kaizeniin yhdistetään usein käsite jonka mukaan suuret tulokset syntyvät pienistä muutoksista ajan myötä. Tämä on kuitenkin toisinaan ymmärretty väärin siten, että kaizen olisi yhtä kuin muutosten vähäisyys. Kyse onkin tuotannon kokonaisvaltaisesta ja jatkuvasta parantamisesta, johon osallistuvat kaikki työntekijät ylimmästä johdosta lähtien. Lean-ajattelun tapaan kaizeniin kuuluu myös ajatus jatkuvasta parannusten tekemisestä sekä turhuuksien vähentämisestä. Kaizen opettaa myös työtehtävien jatkuvaa arviointia ja tarkastelua yksilötasolla. Yksilötason laadunvalvonnassa sovelletaankin usein tieteellistä menetelmää, jonka tarkoituksena on etsiä jatkuvasti ongelmakohtia ja paranneltavia ominaisuuksia. Menetelmänä kaizenia on käytetty monilla muillakin aloilla kuin raskaassa teollisuudessa. Sitä on sovellettu mm. terveydenhuollossa, psykoterapiassa, julkishallinnossa ja pankkitoiminnassa. (Vermeulen 2009.)

Kaizen yhdistetään usein ensimmäisenä autovalmistaja Toyotaan, sekä tämän käyttämään menetelmään, Toyota Production System (TPS). Toyotan menetelmässä tuotannon kaikkien osapuolten odotetaan toimivan sulavasti yhdessä. Ongelmatilanteessa tai minkä tahansa vian ilmetessä tuotantoketju keskeytetään välittömästi, ja henkilökunta pyrkii esimiehensä kanssa löytämään ratkaisun ongelmaan. Ratkaisu tai parannusehdotus pyritään toteuttamaan välittömästi, ja tämän jälkeen toimintaa pyritään havainnoimaan jatkuvasti. Kaizeniin liittyy lähes aina ns. PDCA-sykli (Plan, Do, Check, Act). PDCA-syklissä kierretään nelivaiheista ympyrää, jonka vaiheet ovat suunnittelu (plan), toteutus tai tekeminen (do), tarkistaminen (check) sekä tarvittavien korjausten tai muutosten teko (act). Syklin on tarkoitus toimia eräänlaisena päättymättömänä prosessina, jonka avulla on mahdollista päästä aina hieman lähemmäksi lopullista tavoitetta, usein sitä kuitenkaan saavuttamatta.

Termi kaizen sai alkunsa toisen maailmansodan jälkeen Japanissa, kun johtamistaitojen kehittämiseen erikoistuneelle yksikölle näytettiin opetuselokuva "Improvement in 4 Steps" (Kaizen eno Yon Dankai). Käsitteenä kaizen tuli tunnetuksi maailmanlaajuisesti vasta 1980-luvulla, kun Masaaki Imai julkaisi teoksensa Kaizen: The Key to Japan's Competitive Success, jossa kaizen esiteltiin länsimaiden yrityksille. Teoksessaan Imai luonnehtii kaizenia eräänlaiseksi keinoksi pitää yllä jatkuvaa kehitystä työelämän lisäksi jokaisen omassa henkilökohtaisessa elämässään. (Imai 1986.)

4.2 Lean

”Lean ei ole tila johon pyritään. Se on jatkuva oppimisen ja kehittämisen prosessi” (Tuominen 2010, v). Lean, tai lean-ajattelu, on omanlaisensa filosofia, jossa pääpaino on tuotannon seitsemän eri turhuuden poistamisessa. Tällä on tarkoitus parantaa laatua, asiakastyytyväisyyttä sekä pienentää kustannuksia. Leanin seitsemän turhuutta ovat kuljetukset, varastot, liike, odotusaika, ylituotanto, yliprosessointi sekä viallinen tuote.

Leanin alkuperän yhteydessä mainitaan usein autovalmistaja Toyota Motor Corporation, joka perustettiin jo 1937. Käsite lean kuitenkin syntyi länsimaissa vasta 1980-luvun lopulla. Vuonna 1988 yhdysvaltalainen John F. Krafcik julkaisi artikkelin Lean-tuotantojärjestelmän riemuvoitto (Triumph of the Lean Production System), jossa käsite lean production mainitaan ensimmäisen kerran. Krafcik vertasi artikkelissaan autonvalmistajien eri tuotantojärjestelmiä ja tuottavuustasoja. Krafcikin esittämiä ajatuksia on siitä lähtien kehitetty eteenpäin useampaankin otteeseen, ja voidaan sanoa että viimeistään tähän päivään mennessä leanista on olemassa monia eri määritelmiä. Useille näistä tuntuisi kuitenkin olevan yhteistä pyrkimys tuotannon tehokkuuden lisäämiseen.

Modig ja Åhlström (2013, 5-16) ovat tuoneet esille kahta leaniin liittyvää käsitettä; virtaustehokkuutta ja resurssitehokkuutta. Monessa heidän kuvaamistaan käytännön esimerkeistä pyritään liiaksi resurssitehokkuuteen eli mahdollisimman monen käytössä olevan tuotantoresurssin hyödyntämiseen, ja laiminlyödään virtaustehokkuutta. Virtaustehokkuus voidaan lyhyesti määritellä arvoa tuottavien toimintojen summaksi suhteessa niiden läpimenoaikaan. Kolmas merkillepantava käsite on tässä tapauksessa virtausyksikkö. Teollisessa tuotantoprosessissa virtausyksiköllä tarkoitetaan jotakin valmistettavaa tuotetta, joka käy läpi valmistus-, sekä jalostusprosessin, kun taas esimerkiksi palvelualalla virtausyksikkö tarkoittaa ihmistä. Siinä missä teollisessa prosessissa tuotetta jalostetaan erilaisilla materiaaleilla, pyritään palveluprosessissa puolestaan täyttämään ihmisen tarpeita eri toiminnoilla. Ihannetapauksessa olisikin tavoitteena toteuttaa prosessissa virtaustehokkuutta, hylkäämättä kuitenkaan samaan aikaan resurssitehokkuutta.

Vaikka leanin toimintaperiaate on käyttökelpoinen monessa valvotussa menetelmässä, ei leanin menetelmiä voi sellaisinaan soveltaa ohjelmiston kehittämiseen tai ohjelmistosuunnitteluun. Hyvän ohjelmiston suunnitteleminen ei ole tuotantoprosessi (production process), vaan se on kehitysprosessi (development process). (Väyrynen 2009, 19.) Tässä yhteydessä voidaankin mainita vielä Tom, ja Mary Poppendieckin vuonna 2007 kehittämä lean-ohjelmistokehitys (lean software development). Vaikka tätä ennen oli ollut jo olemassa käsite ketterästä ohjelmistokehityksestä (agile software development), vei lean-ohjelmistokehitys osaltaan tämän idean vielä hieman pidemmälle.

Aiempien määritelmien tapaan myös lean-ohjelmistokehityksessä määritellään seitsemän pääperiaatetta. Nämä periaatteet ovat turhuuden poistaminen, laadun rakentaminen, tietoisuuden lisääminen, sitoutumisen lykkääminen, nopea toimitus, ihmisten kunnioitus ja kokonaisuuden optimointi.

Ensimmäinen periaatteista viittaa leanin jo aiempana mainittuun ajatukseen seitsemän eri turhuuden poistamisesta. Laadun rakentamisella tarkoitetaan yksinkertaisesti laadukkaan ohjelmakoodin tuottamista alusta alkaen. Tietoisuuden lisäämisellä tarkoitetaan tiettyä selkeyttä ja läpinäkyvyyttä, joka syntyy kun kaikki työntekijät tietävät tarkalleen, mitä milloinkin on tarkoitus tehdä. Sitoutumisen lykkääminen voi olla terminä hieman harhaanjohtava. Sillä tarkoitetaan sellaisten päätösten ja valintojen siirtämistä myöhemmäksi, joilla on peruuttamattomia vaikutuksia prosessin suhteen. Nopealla toimituksella tarkoitetaan paitsi kirjaimellisesti tuotteen eli ohjelmiston nopeaa valmiiksi saattamista, niin etenkin nopeaa reagointia tilanteisiin sekä jatkuvaa laadunvalvontaa. ”It is impossible to deliver fast without a very high level of quality” (Noronen 2014, 12).

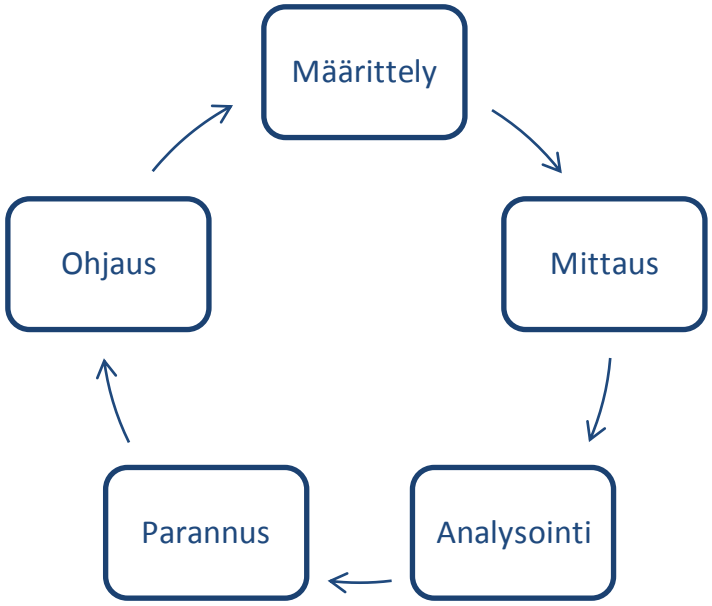
Ihmisten kunnioitus viittaa periaatteeseen, joka on ollut läsnä lean-ajattelussa jo Toyotan ajoilta lähtien. Ihmisten välinen vuorovaikutus ja toistensa kunnioitus luo mielekkäämmän ja ennen kaikkea onnellisemman työympäristön, jonka katsotaan olevan edellytyksenä tehokkaammalle toiminnalle. Kokonaisuuden optimoinnilla tarkoitetaan jatkuvaa kokonaisuuden tarkastelua sen sijaan, että keskityttäisiin ns. mikrotason ominaisuuksien parantamiseen.

4.3 Six Sigma

Six Sigma on laatujohtamisen työkalu joka perustuu tilastotieteeseen. Siinä pyrkimyksenä on mitata prosessin virheiden määrää, ja poistaa sen jälkeen nämä virheet systemaattisesti. Six Sigman kehitti alun perin insinööri Bill Smith vuonna 1986, ollessaan töissä Motorola -suuryrityksessä. Asiakaskeskeisyyttä pidetään olennaisena piirteenä six sigmassa, sillä asiakastyytyväisyys määrittää lopulta sen kuinka hyvin tuotteen kehittämisessä ollaan onnistuttu.

Six sigmassa käytetään ns. DMAIC-menetelmää (kuvio 1), joka on jatkuva syklinen prosessi. DMAIC on lyhennys sanoista define, measure, analyze, improve, control. Jokainen näistä käsittää yhden vaiheen itse prosessissa. Ensimmäinen vaihe eli määrittely tarkoittaa parannustarpeiden kattavaa kartoitusta, jossa pyritään tunnistamaan myös asiakkaan vaatimukset. Mittaus tarkoittaa vaihetta jossa olemassa olevaan prosessiin liittyvä tieto ensin valmistellaan ja sitten kerätään, jotta saataisiin selkeämpi käsitys suorituskyvystä. Analysointivaiheessa arvioidaan kerätyn tiedon ja määrittelyiden perusteella mahdollisia ongelmakohtia. Parannus on vaihe jossa suunniteltu parannus varsinaisesti toteutetaan käytännössä. Parannusvaiheen toteutusta valvoo usein projektijohtaja, ja monesti tällaisen menettelyn katsotaan vähentävän epäonnistumisen riskejä. Varsinkin jos toteutettava projekti on mittakaavaltaan erityisen suuri, tai sisältää merkittävän muutoksen olemassa olevaan tuotteeseen, on riskien välttäminen tärkeää. Käsitteellä ohjaus on six sigmassa kaksi eri merkitystä. Ensimmäinen pyrkii vastaamaan kysymykseen milloin tehdä jokin asia, mutta myös siihen mitä tehdä, milloin ja kuinka paljon. Toinen merkitys painottaa milloin on oltava tekemättä mitään. (Neuendorf 2004, 29-31.)

Six sigma yhdistettiin vuonna 2002 leanin kanssa, ja syntynyttä uudistusta kutsutaan nimellä Lean Six Sigma.



Kuvio 2: DMAIC-menetelmä

5 Järjestelmän suunnittelu ja toteutus

Ennen projektin alkamista oli kuvapankkipalvelulle tehty arvio erilaisten vaihtoehtojen toteutus-, ja ylläpitokustannuksista (liite 2). Tämän lisäksi pidettiin tilastoa kaikista yhtiön omien myyntituotteiden kuvatiedostoista, ja näiden vuosittaisesta kertymästä (taulukko 2). Kuten kertymätaulukosta voidaan nähdä, ei yksittäisten kuvatiedostojen määrä ollut seurannan alkuvaiheessa mitenkään erityisen suuri. Voidaan myös huomata, että lisättyjen uusien kuvien määrä on ollut käytännössä koko seurannan ajan jatkuvassa kasvussa. Uusien kuvien jatkuva lisääminen palvelimelle tulisi näin lopulta johtamaan siihen, että kuvatiedostojen todellinen määrä yltäisi varsin pian todella suurin lukemiin. Viimeistään tähän päivään mennessä voidaan jo puhua miljoonista yksittäisistä kuvista palvelimella.

Vuosi	Kuvia
2011	10845
2010	11604
2009	10979
2008	8833
2007	6926
2006	5549
2005	4512
2004	4402
2003	3286
2002	2652
2001	227
2000	7
Yhteensä	69822

Taulukko 2: Kuvien kertymä

Projektin suunnitteluvaiheessa viikoilla 26 ja 27 (ks. taulukko 1) hahmoteltiin ensin järjestelmän rakenne sekä tietokanta. Samalla rajattiin tarkkaan mitä ominaisuuksia haluttiin, ja minua pyydettiin myös usein miettimään jo ennalta eri keinoja joilla toteuttaisin näitä ominaisuuksia.

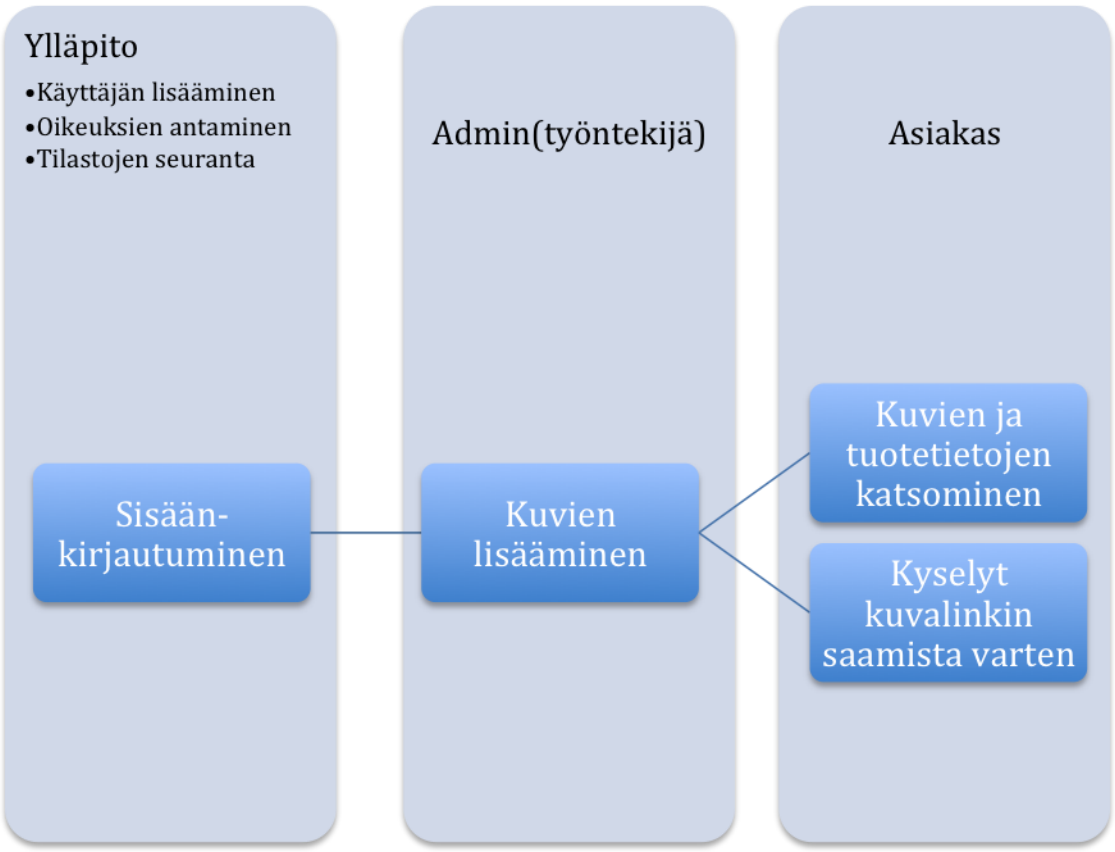
Projektille oli tehtävä tekniset määrittelyt ennen kuvapankkipalvelun toteuttamista. Käytännössä tämä tarkoitti sitä, että keräsin tehtyjen suunnitelmien pohjalta kaikki kartoitetut ominaisuudet sekä tavoitteet, ja tein näistä oman dokumentin. Määrittely-dokumentissa on ku-

vattuna kaikki kuvapankin toiminnallisuudet mahdollisimman tarkasti. Dokumentointi paitsi auttaa hahmottamaan järjestelmän rakenteen paremmin, on siitä myös hyötyä projektin ollessa jo käynnissä.

Järjestelmässä olevien tuotteiden päivitys tapahtuu ajastettujen toimintojen avulla. Tietoja ei kuitenkaan päivitetä Ruotsista suoraan BTJ Finlandin omalle palvelimelle Lauttasaareen, vaan ensiksi tuotetiedot siirretään Ruotsista käsin Tampereella sijaitsevalle palvelimelle, johon kenelläkään yksittäisellä käyttäjällä ei ole pääsyä. Pääsy ns. pääpalvelimelle on estetty yksinkertaisesti tietoturvasyistä. Tämän jälkeen tiedot päivittyvät edelleen erityiselle Replika-palvelimelle, jonka ainoa tarkoitus on vain kopioida kaikki varsinaisen tuotetietokannan tietueet. Replika toimii pääpalvelimen varmuuskopiona, jolle tosin tässä tapauksessa tehdään varsinaiset muutokset, mikäli tarpeen. Replikalta tiedot siirtyvät lopulta Lauttasaaren toimiston omalle palvelimelle. Replikalle on mahdollista päästä normaalisti käsittelemään tuotetietoja. Tiedon siirtyminen usean eri palvelimen kautta auttaa varmistamaan, että tiedot ovat kulloinkin oikeat. Tarvittaessa palvelimien sisältöjä voidaan esimerkiksi verrata keskenään, mikäli tietokannassa ilmenee vikoja.

Järjestelmän kehitystä varten perustettiin oma palvelin, jolle kaikilla tiimin jäsenillä oli vapaa pääsy. Tiedostoja oli mahdollista siirtää palvelimen ja eri käyttäjien välillä. Palvelimelle oli mahdollista tehdä muutoksia ottamalla siihen yhteys komentorivipohjaisella käyttöliittymällä. Myös kehityspalvelimella hyödynnettiin ajastettujen toimintojen käyttömahdollisuutta.

Oma osuuteni käsitti tarkemmin ottaen kyselyrajapinnan kehittämisen sekä visuaalisen käyttöliittymän toteuttamisen tuotantoa varten. Tällä hetkellä kyselyrajapinta on Helsingin kaupunginkirjastojen käytössä. Käyttöliittymä puolestaan oli lähtökohtaisesti pelkästään yhtiön omaan käyttöön. Kuvapankkiin suunniteltiin toteutettavaksi lisäksi erilaisia käyttöoikeustasoja (kuvio 3). Kyselyrajapintaa käytetään tuotetietojen ja kuvien hakemiseen asiakkaiden käyttöön.



Kuvio 3: Käyttöoikeustasot

Kaikessa yksinkertaisuudessa kyselyrajapinta käsitti joukon parametreja, joiden avulla tuotteen kuva tai tiedot tuotiin näkyviin. Pakollisia parametreja oli kaksi; asiakkaan tunniste eli id sekä tuotteen tunniste. Valinnaisilla parametreilla pystyttiin esimerkiksi määrittämään kuvakoko, ja haluttaessa voitiin tuoda näkyviin pelkästään tuotteen tiedot ilman kuvaa. Valinnaisilla parametreilla oli kullakin oma oletusarvo, mikäli näille ei erikseen asetettaisi mitään arvoa kyselyn yhteydessä.

Kyselyrajapinnan oli myös sisällettävä ominaisuuksia, jotka estäisivät kuvien katsomisen, mikäli asiakkaan tarvittavat oikeudet puuttuisivat. Vastaavasti asiakkaalle tulostettaisiin ilmoitus, mikäli hänen haluamalleen tuotteelle ei olisi saatavilla olemassa olevaa kuvaa.
Jokaisella asiakkaalla oli määritetty tietokantaan oikeudet vain niihin tuotteisiin, joista tämä itse oli sopinut. Tietokannassa tuotteiden oikeudet riippuivat normaalisti näiden kustantajasta ja asiakkaat olivat varsin tietoisia siitä, miltä kustantajalta he tuotteitaan hankkivat.

6 CI-mallin soveltaminen järjestelmän kehittämisessä

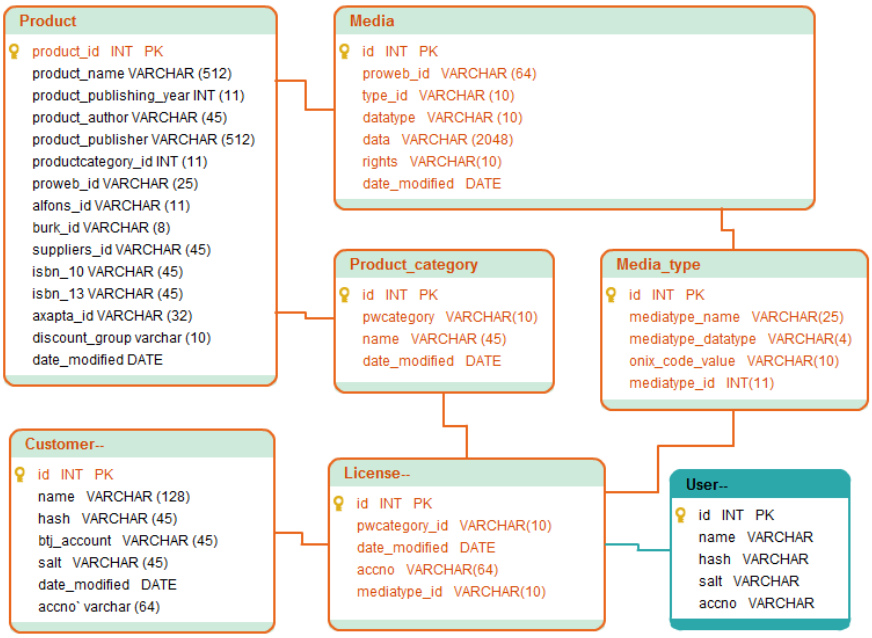
Armas-kuvapankkipalvelu kehitettiin käyttämällä jatkuvan kehittämisen mallia. Käytännössä tämä tarkoitti jatkuvaa tarpeiden ja toiminnallisuuksien kartoittamista sekä testausta. Edellä tulen kertomaan tarkemmin etenkin kyselyrajapinnan ja tämän ohjelmakoodiin tehdyistä muutoksista. Samalla tavoin kuvapankin tietokantaa muutettiin projektin aikana useampaankin otteeseen. Tietokannan ja kyselyrajapinnan lisäksi käsittelen tuotantoprosessin tukea, eli kuvatuotannon graafista käyttöliittymää sekä kehittämiseen käytettyä palvelinympäristö. Projektin aikana käytetty kehitysympäristö, jossa ohjelmakoodia tavallisesti testattiin ennen tuotantoon laittamista, voidaan nähdä yhdenlaisena sovelluksena aiemmin mainitusta PDCA-syklistä.

Kehitysympäristöön siirrettiin jatkuvasti uusia versioita jo olemassa olevasta koodista. Samaan aikaan kukin tiimin jäsenistä yleensä testasi ohjelmistoa omalta työasemaltaan, ja antoi omia ehdotuksia muutoksia ja parannuksia varten. Käytössämme ei ollut mitään varsinaista versionhallintajärjestelmää mahdollisesti siksi, että projektin alussa käsiteltävien tiedostojen määrä palvelimella ei ollut erityisen suuri. Tavallisesti esimiehemme kertoi aina ensimmäisenä viikkopalavereissa, minkälaisia uusia ominaisuuksia järjestelmään ensisijaisesti haluttiin. Tarvittaessa yhteydenpitoon saatettiin käyttää sähköpostia, joka mahdollisti esimerkiksi linkin lähettämisen ohjelmiston esittelyä ja testausta varten. Ohjelmistokoodia oli etäyhteyden avulla mahdollista tarkastella kehityspalvelimelta erikseen sellaisenaan. Ohjelmistokoodin ohella palvelimelle saatettiin lisätä muutakin mahdollista aineistoa. Esimerkiksi kyselyrajapinnan tuotantovaiheessa tarvittiin valmiin ’ei kuvaa’ –kuvan lisäksi ns. ’kuvaa ei saatavilla’ –kuva. Tätä oli tarkoitus käyttää tilanteessa, jossa asiakas yrittäisi hakea kuvaa tuotteelle johon hänellä ei ole oikeuksia. Myös tuotetietosivua varten oli tehtävä ulkoasun hienosäätöä CSS-määrittelyiden avulla. Tästä puolesta vastasi pääasiassa eräs toinen tiimin jäsenistä, esimiehen määräyksestä, sillä CSS-määrittelyiden toteutus ja testaus voi usein olla varsin aikaavievää.

6.1 Tietokanta

Kuten aiempana on kerrottu, tehtiin kuvapankkipalveluun jo suunnitteluvaiheessa hahmotelma tietokannan rakenteesta. Viimeistään kyselyrajapinnan ensimmäistä versiota varten oli luotava valmis tietokanta tauluineen. Tauluja muutettiin sekä täydennettiin lukuisia kertoja projektin aikana lisäämällä näihin kenttiä, kunnes voitiin puhua tietokannan lopullisesta muodosta (kuvio 4). Tietokantaan tallentuu paitsi olemassa olevien tuotteiden eri mediat ja näi-

den omat tiedot, niin myöskin tiedot kaikista ylläpidon käyttäjistä sekä asiakkaista. Tuotteisiin liittyviä tietokantatauluja on yhteensä viisi; ”license”, ”media”, ”media_type”, ”product” ja ”product_category”. Näiden lisäksi luotiin vielä taulut ”customer”, ja ”user”. Taulut viittasivat toisiinsa tietokanta-avainten avulla. Esimerkiksi taulussa ”media_type” on avain ”mediatype_id”, jonka avulla taulu ”media” voi viitata siihen. Taulun ”media” lisäksi ”license” -taulussa oli viiteavain media_type -tauluun. Lisenssitaulussa on viiteavaimet edelleen sekä asiakas-, käyttäjä-, että tuotekategoriatauluun (”product_category”). Tietokantaan kuitenkin tehtiin lukuisia lisäyksiä ja muutoksia sekä tuotantovaiheen aikana että tätä ennen. Tietokantaan esimerkiksi lisättiin tuotantovaiheessa taulut ”productcategory” ja ”media_type”.



Kuvio 4: Tietokantarakenne

Kuvapankin oli kyettävä paitsi jaottelemaan eri tuotteet tietoineen, oli sen myös jaoteltava kaikki tuotteisiin liittyvä media. Tietokannan taulu ”media” sisälsi tuotteiden olemassa olevaa mediaa, jota olivat vielä tuotantovaiheen alussa tuotteiden kuvat sekä esittelytekstit. Taulussa tämä oli toteutettu niin, että jos tuotteelle oli olemassa yksi tai useampi kuva, oli näistä jokaiselle ensinnäkin oma viittaus taulun ”media_type” kentässä ”mediatype_datatype”. Tämän lisäksi varsinainen mediasisältö kirjattiin ”media” -taulun kenttään ”data”. Kentän maksimipituudeksi määriteltiin 2048 merkkiä. Kuviin viitattiin yksinkertaisesti kirjaamalla näiden oikea tiedostonnimi ”data” -kenttään, johon viittaamalla kyselyrajapinta löysi palvelimelta samannimisen kuvatiedoston. Vastaavasti esittelyteksti voitiin kirjata kokonaisuudessaan tietokantataulun kenttään. Näin ollen kyselylauseen avulla oli mahdollista päästä käsiksi tuotteiden mediaan ottamalla ensin yhteys tietokantaan halutun tuotteen id:llä, johon media-

taulussa oli myös oma avainkenttänsä. Saatua kyselytulosta hyödynnettiin edelleen ohjelmakoodissa median tulostamisessa asiakkaan näkyviin.

Havaittiin kuitenkin, että tuotetietojen ja tiedostojen lisäksi mediaakin olisi tarpeellista pystyä jaottelemaan. Yhtenä syynä tähän oli että kyselyjä kyettäisiin tällöin hallitsemaan paremmin. Ohjelmisto toimii sitä nopeammin, mitä pienemmän rivimäärän kyselylause joutuisi käymään läpi jokaisen tietokantahaun kohdalla. Jaottelemalla sisältöä oli mahdollista pienentää jokaisen kyselylauseen kohdalla läpikäytävien tietueiden määrää. Varsinkaan kahden erityyppisen median sisältöjä ei olisi toivottavaa saada saman kyselylauseen tuloksiksi. Ratkaisuna tähän oli taulu media_type. Taulu sisälsi seuraavat sarakkeet: id (pääavain), mediatype_name, mediatype_datatype, onix_code_value sekä mediatype_id. Media_type jakoi eri mediat näiden tietotyypin mukaan, antamalla näille aina arvoksi joko ”TEXT” tai ”FILE”. Molemmille mediatyypeille oli olemassa oma id, jolla viitattiin tarkemmin median sisältöön. Esittelytekstin tapauksessa id oli nimeltään ”DESC”, kuvan id taas oli ”NCO”.

Taulu productcategory (kuvio 6) sisältää kaikki BTJ:n olemassa olevat tuotekategoriat, joita oli lisäyshetkellä yli sata. Näihin kuuluivat multimedian ja audiovisuaalisen aineiston lisäksi esimerkiksi erilaiset kalusteet, oppikirjat ja tarvikkeet. Tämän taulun tarkoitus oli yksinkertaisesti selkeyttää kuvapankin sisältöä integroimalla jo olemassa oleva listarakenne osaksi kuvapankkia. Vastaavasti tuotteiden kategorioita on mahdollista käyttää kyselyjen hakuehtoina. Lisäksi olemassa oleviin tauluihin saatettiin lisätä avaimia, joiden oli tarkoitus parantaa järjestelmän toimivuutta.

```
mysql> select * from productcategory;
+-----+------------+---------------------------------+---------------+
| id  | pwcategory | name                            | date_modified |
+-----+------------+---------------------------------+---------------+
|  80 | 000        | Konversion perustama            | NULL          |
|  81 | 001        | Kalusteet (irto)                | NULL          |
|  82 | 002        | Koulukalusteet                  | NULL          |
|  83 | 002??      | 002??                           | NULL          |
|  84 | 002x       | 002X                            | NULL          |
|  85 | 006??      | 006??                           | NULL          |
|  86 | 007        | Projektit                       | NULL          |
|  87 | 008        | Toimistokalusteet               | NULL          |
|  88 | 010??      | 010??                           | NULL          |
|  89 | 011        | Painotuotteet                   | NULL          |
|  90 | 012        | Kirjamuovit                     | NULL          |
|  91 | 013        | Tarvikkeet                      | NULL          |
|  92 | 015??      | 015??                           | NULL          |
|  93 | 019        | Toimituskulut                   | NULL          |
|  94 | 020        | Suuret kustantajat              | NULL          |
|  95 | 021        | Muut kustantajat                | NULL          |
|  96 | 022        | Sidotut kirjat                  | NULL          |
|  97 | 023        | Oppikirjat                      | NULL          |
|  98 | 024        | Erikoiskirjat                   | NULL          |
|  99 | 025        | Ruotsalaiset kirjat             | NULL          |
```

Kuvio 5: Tuotekategoriat

Tietokanta sisälsi omat taulut asiakas- ja käyttäjätileille. Molempiin tauluihin liittyvät salasanat tallennettiin tietokantaan käyttämällä tiivistemenetelmää (englanniksi hash). Käytetty tiivistealgoritmi oli tässä tietokannassa nimeltään MD5. Muita vastaavia algoritmeja ovat esimerkiksi SHA-1 ja CRC. Tiivistemenetelmässä ennalta määritelty tiivistealgoritmi muuntaa annetun salasanan merkkijonoksi, jonka pituus on MD5-algoritmin tapauksessa 32 merkkiä. Tiiviste on yksisuuntainen, eli sitä ei ole mahdollista muuntaa takaisin alkuperäiseksi merkkijonoksi. Tämä vähentää tietoturvariskejä, sillä tietomurron sattuessa ei tietokantaa tallennettuja alkuperäisiä salasanoja ole mahdollista saada selville. Sisään kirjautumisen aikana ohjelmisto luo syötetystä salasanasta samalla tavoin tiivisteen, ja vertaa tätä tietokannassa olevaan oikean salasanan tiivisteeseen. Mikäli nämä kaksi täsmäävät, kirjataan käyttäjä sisään. Kyseinen menetelmä on varsin yleinen monessa järjestelmässä, joissa hallinnoidaan suurta määrää käyttäjätunnuksia ja salasanoja.

Tiivistemenetelmän lisäksi puhuttiin usein mahdollisuudesta käyttää ns. salt-menetelmää. Salt on kryptografiassa käytetty termi, jolla tarkoitetaan yleensä satunnaisesti luotua merkkijonoa, joka lisätään alkuperäiseen syötteeseen, kuten käyttäjien ja asiakkaiden kirjoittamiin salasanoihin. Saltin tarkoitus on lisätä tietoturvaa muuttamalla tiivistealgoritmin käsittelemää merkkijonoa, sekä sitä kautta lopullista tiivistemerkkijonoa, ja näin hankaloittaa tietomurtoja. Menetelmän toteuttamiseksi minua kehotettiin luomaan jo ennalta tietokantaan valmiit kentät salt-arvoja varten, ja miettimään keinoja toiminnallisuuden toteuttamiseen. Tarkoituksena oli, että satunnaisluomisen lisäksi ylläpidon olisi mahdollista halutessaan muuttaa tai lisätä tunnuksien salt-arvoja käsin.

Usein tietokannan toimivuutta ja kyselylauseita testattiin suoraan palvelimelta käyttämällä komentorivipohjaista käyttöliittymää. Tämän avulla oli helpompi hahmottaa tietokannan rakennetta ja toimintaa ohjelmistokehitystä silmällä pitäen. Monesti saatoinkin kokeilla ensin komentoriviltä samoja kyselylauseita, joita käytin ohjelmakoodissa, jotta pystyin varmistumaan siitä että koodini toimisi moitteetta. Vastaavasti ongelmatilanteissa oli kätevää tarkistaa komentorivin kautta, johtuiko ohjelmiston toimimattomuus mahdollisesti sittenkin siitä, ettei tietokannasta löytynyt haluttua dataa.

6.2 Palvelinympäristö

Ennen kuvapankin kyselyrajapinnan siirtämistä tuotantopalvelimelle järjestelmään oli tehtävä tietyt pakolliset perusominaisuudet, joita on kuvailtu jo aiemmin. Tavallisesti etenkin ohjelmistotuotannon kaikkiin vaiheisiin liittyy laadunvarmistustoimenpiteitä, joilla pyritään kitkemään virheitä mahdollisimman varhaisessa vaiheessa (Haikala & Märijärvi 2004, 37). Vaikka

tässä tapauksessa kyse oli kehitysprosessista, pyrittiin testaamista suorittamaan samalla tavoin alusta asti.

Projektin alkuvaiheessa kyselyrajapinnan kehittämiseen ei ollut saatavilla vielä muita resursseja ja apuvälineitä kuin omalta työasemaltani löytyvät ohjelmistot sekä kehityspalavereissa valmistellut suunnitelmat ja määritykset. Tämän takia minun oli tehtävä ensimmäinen versio tietokantatauluista ja kyselyrajapinnan ohjelmakoodista omalle pöytätietokoneelleni. Perustason toimintojen testaus toki toimi työpöytäympäristössä, koska käsiteltävä tietomäärä oli ensimmäistä versiota tehtäessä vielä varsin pieni. Syötin tietokantaan ensin manuaalisesti itse tehtyä testidataa, jonka avulla pystyin ajamaan ja testaamaan kyselyrajapinnan toimintoja. Myöhemmin sain yhdeltä tiimin jäsenistä käyttööni testidataa, joka oli kopioitu varsinaisesta tuotetietojärjestelmästä. Oikealla testidatalla pystyttiin näin ollen simuloimaan varsinaisen kuvapankin sisältöä ja toimintaa. Pidemmän päälle työpöytäympäristö oli kuitenkin palvelinkäyttöisen ohjelmiston testaukseen varsin hidas. Myöskään suuremman datamäärän käsittely ei onnistuisi, vaan tuottaisi suurella varmuudella ongelmia kevyellä laitteistolla, joten tehokkaammalle palvelimelle oli selkeästi käyttöä.

Ennen pitkää kyselyrajapinnan ja kuvapankin muidenkin toimintojen kehitystä varten perustettiin oma FTP-palvelin. FTP on lyhennys sanoista File Transfer Protocol, joka tarkoittaa kahden tietokoneen välistä tiedostonsiirtomenetelmää. FTP-yhteys toimii aina asiakas-palvelin –periaatteella, jossa asiakas eli käyttäjä ottaa yhteyden palvelimeen. Kehityspalvelin käytti CentOS-käyttöjärjestelmää (lyhennys sanoista Community Enterprise Operating System). CentOS on avoimen lähdekoodin Linux-käyttöjärjestelmän jakelupaketti, joka pohjautuu kaupalliseen Red Hat Enterprise Linuxiin (RHEL). Syynä vapaan lähdekoodin palvelinohjelmiston käyttöön kaupallisen sijaan oli ensisijaisesti joustavuus. Linux-palvelimella oli mahdollista suorittaa tiettyjä ajastettuja toimintoja, joita ei ollut mahdollista saada toimimaan Microsoftin vastaavalla kaupallisella palvelinympäristöllä. Lisäksi Linux-palvelinohjelmisto oli osoittautunut jo aikaisemmassa käytössä Microsoftin vastaavaa vakaammaksi.

Kehityspalvelimella eri toiminnallisuuksien testaaminen oli paitsi nopeampaa, myös kätevämpää, sillä palvelimeen pystyi ottamaan yhteyden kuka tahansa, jolla vain oli pääsy yhtiön sisäiseen langattomaan verkkoon ja palvelimen IP-osoite. Toimintaperiaate olikin jossain määrin sama, kuin ohjelmistokehityksessä usein käytetyssä ohjelmiston versionhallinnassa. Mikäli esimieheni esimerkiksi halusi nähdä kyselyrajapinnan uusimpaan versioon toteutetut muutokset toiminnassa, kykeni hän ottamaan yhteyden palvelimeen omalta työpisteeltään. Saatuamme kuvapankin ensimmäisen version laitettua tuotantopalvelimelle, aloimme saada palautetta sitä käyttäviltä asiakkailta. Palautteen mukana saimme välillä myös parannusehdotuksia uusia ominaisuuksia varten. Esimerkiksi kyselyrajapinnan virheparametri oli alun perin asiakkaan toivoma lisäominaisuus.

Muut tiimin jäsenet saattoivat lisäksi seurata ohjelmistoon tehtyjä muutoksia ja ehdottaa parannuksia. Tarvittavien muutosten ja parannusten, sekä näiden testaamisen, jälkeen ohjelmisto siirrettiin yleensä kehityspalvelimelta tuotantoon. Kyselyrajapinnan kohdalla varsinainen kehitys oli kuitenkin minun vastuullani, joten muutamaa poikkeusta lukuun ottamatta toiset tiimimme jäsenet eivät tehneet muutoksia itse ohjelmakoodiin. Olen jo aiemmin kertonut projektin suunnittelusta ja sitä varten tehdyistä määrittelyistä. Samaan tapaan sovimme aikataulun eri vaiheineen ennalta niin että kaikille olisi mahdollisimman selvää, mitä milloinkin oli tarkoitus tehdä. (Kim 2012.)

6.3 Kyselyrajapinta

Teoriassa kuvapankkiprojektin oli tarkoitus edetä taulukon 1. mukaisen viikkoaikataulun mukaisesti, mutta käytännössä joistakin vaiheista ja ominaisuuksista kuitenkin hieman viivästyttiin. Perinteistä projektityöskentelyä ketterämmän menetelmän käyttö piti tällaisissa tilanteissa hyvin huolen siitä, ettei projektin eteneminen vaarantunut merkittävästi. Käytännössä ketterä menettely tarkoitti tässä sitä, että projektin resursseja käytettiin tarvittaessa paikkaamaan viivästyksiä. Esimerkiksi kun kyselyrajapinnan siirtäminen tuotantoon viivästyi hieman sovitusta ajankohdasta, oli toisen tiimimme jäsenen autettava ohjelmakoodin viimeistelyssä.

Kuten aiemmissa luvuissa on kerrottu, kyselyrajapinta ottaa vastaan arvoja parametreina. Koska rajapinta ei itsessään sisällä graafista käyttöliittymää, annetaan parametrit kirjoittamalla halutut arvot näille varattuihin kohtiin selaimen osoiteriville. Kun parametreille on määritelty arvo, luo ohjelmakoodi siitä kyselylauseen jolla tarkistetaan löytyykö tietokannasta annettua arvoa vastaavaa tuotetta. Pakollisia parametreja on yhteensä kaksi; ”id” eli asiakkaan oma tunniste ja ”pid” eli tuotteen tunniste.

Alun perin kyselyrajapinnan oli tarkoitus tulostaa oletuksena pelkkä kuva. Monilla asiakkaista on todennäköisesti eniten käyttöä juuri tuotteiden kuvatiedostoille sivuillaan. Tämä ominaisuus toteutettiin käyttäjätasolla niin, että jätettäessä antamatta parametrille ”qtype” mitään arvoa, määritteli ohjelmakoodi itse tälle arvon ”m”, joka taas sai aikaan sen että pelkkä kuva tulostui selaimeen. Asettelemalla sivun lähdekoodiin halutuilla parametreilla varustetun linkin itse kyselyrajapintaan, on asiakkaan mahdollista upottaa tuotteen kuva minne tahansa omalla sivustollaan. Helsingin kaupunginkirjastot hyödyntää upotteita esimerkiksi pienoiskuvina omalla sivustollaan (kuvio 6). Pienkuva eli thumbnail sisältää tässä tapauksessa myös linkin tuotetietosivulle. Kirjain ”m” viittaa sanaan media, ja tässä tapauksessa medialla tarkoitetaan kuvatiedostoa.

Kuvio 6: Upotelinkki

Ohjelmointitasolla toiminnallisuus toteutettiin käyttämällä valmista funktiota nimeltä SimpleImage. Funktio on saatavilla ns. vapaana ohjelmistona, eli kuka tahansa saa käyttää, kopioida, muokata ja jakaa ohjelmistoa edelleen vapaasti. Funktio avaa ensin halutun kuvatiedoston ja tallentaa tämän hetkeksi välimuistiin. Avatessa ja tallentaessa tiedoston funktio tallentaa myös kuvatiedoston alkuperäiset mitat sekä tiedostotyypin (JPG, GIF tai PNG). Tämän jälkeen funktio yksinkertaisesti tulostaa näkyviin pelkän kuvan, toisin sanoen funktio ei viittaa enää itse tiedostoon siinä kohti kun haluttu kuva on tulostettu näkyviin. Aiemmin kuvan tulostamiseen käytettiin PHP:n omaa fopen-funktiota, mutta tämän avulla esimerkiksi kuvakoon muuttaminen oli hankalampaa. SimpleImage puolestaan sisälsi valmiiksi toiminnot kuvakoon muuttamiseen eri tavoin kuten skaalaamalla. Kuvan tulostus oli pyydetty toteuttamaan tällä tavoin nimenomaan tietoturvan kannalta. Kun rajapinta tulostaa pelkän kuvan, sen sijaan että viittaisi tavalliseen tapaan suoraan alkuperäiseen tiedostoon palvelimella, ei sivun lähdekoodista ole käytännössä mahdollista nähdä edes kuvatiedoston nimeä tai sijaintia.

Kyselyrajapinnalla oli myös mahdollista tulostaa pelkkä esittelyteksti. Tämä tapahtui antamalla valinnaisparametrin arvoksi ”t”, jossa kirjainlyhenne ”t” viittaa luonnollisesti tekstiin. Neljäs vaihtoehto oli arvo ”xml”, joka tulosti xml-listauksen kaikesta valittuun tuotteeseen liittyvästä mediasta. Tämä vaihtoehto kehitettiin alun perin ainoastaan ylläpidon ja kuvatuotannon käyttöön heidän pyynnöstään.

Parametri ”qtype” oli ensimmäinen ns. valinnaisparametri. Muita valinnaisparametreja olivat ”ftype”, ”format”, ”debug” ja ”error”. Format ja debug suunniteltiin etupäässä kehityskäyttöä varten, eikä niitä ole näin ollen saatavilla tuotantoversiossa. Parametri ”ftype” mahdollistaa kuvakoon muuttamisen silloin, kun kyselytyyppinä olisi kuva eli parametri ”qtype” saisi arvon ”m”. Kun arvoa ei erikseen määritellä, saa parametri ”ftype” oletusarvon ”04”, joka tulostaa mitoiltaan vakiomääritellyn kansikuvan. Kansikuvalle määriteltiin vakiona leveydeksi 240 pikseliä. Kyselyrajapinnan ohjelmakoodi piti huolen siitä että kuvan alkuperäissuhde säilyi

vaikka kuvakoko muuttuisi. Muut valittavat arvot ovat ”06” eli thumbnail, joka tulostaa pienen kuvan. Thumbnailin vakiomääritelty leveys oli 130 pikseliä. Näiden lisäksi oli saatavilla vielä arvo ”07”, joka tulostaa kuvan alkuperäisessä koossaan. Jos kuvan alkuperäiskoko on leveydeltään pienempi tai yhtä suuri kuin kansikuvan 240 pikseliä, ei arvon ”07” asettaminen parametrille ”ftype” tulosta suurempaa kuvaa, vaan mitat vastaavat kansikuvaa (”04”).

Parametri debug oli nimensä mukaisesti tarkoitettu kehityksen seurannan apuvälineeksi. Sille on mahdollista antaa joko arvo ”0” tai ”1”. Oletuksena debug saa aina arvon 0, mutta jos arvoksi asetetaan 1, tallentaa kyselyrajapinta kaikki sille annetut ilmoitukset palvelimen lokiin. Tästä oli hyötyä varsinkin myöhemmässä kehitysvaiheessa. Viimeinen eli virheparametri, saa normaalisti oletuksena arvon ”0”. Tällöin rajapinta tulostaa asiakkaalle ns. ”ei kuvaa”-kuvan, mikäli halutulle tuotteelle ei löydy olemassa olevaa kuvaa. Asiakas voi kuitenkin halutessaan antaa virheparametrille arvon ”1”, jolloin rajapinta tulostaa näkyviin normaalin virheilmoituksen. Virheilmoitus tallentuu luonnollisesti myös palvelimen lokiin tilastoseurantaa varten.

Ohjelmakoodissa oli aluksi vajaavaisuuksia liittyen ominaisuuteen, joka tulostaisi ns. tuotetietosivun. Alunperin tämä toiminnallisuus oli toteutettu kokonaan omassa tiedostossaan, johon varsinainen kyselyrajapinta otti yhteyden antamalla tarvittaville muuttujille parametreina saadut arvot. Erillinen tiedosto oli oikeastaan HTML-dokumentti, johon vain lisättiin tarvittavat muuttujat PHP:n avulla. Kyselyrajapinnan oli kuitenkin kyettävä tarkistamaan asiakkaan käyttöoikeudet kaikissa kyselyissä, joten tuotetietosivun toteuttaminen erillisestä tiedostosta käsin hankaloitti lopulta jonkin verran järjestelmän toimintaa. Lisäksi molemmissa tiedostoissa oli oltava oma kuvanhakutoiminto. Alkuvaiheessa tätä kokeiltiin toteuttaa jopa niin, että tuotetietosivu käyttäisi käskyn saatuaan jälleen kyselyrajapinta-tiedostoa kuvan hakuun. Tämän tyylinen menettely, paitsi hidasti ohjelmistoa, sisälsi myös mahdollisen riskin ohjelmiston joutumisesta ns. ikuiseen silmukkaan.

Lopulta toiminnallisuus päätettiin toteuttaa PHP:n avulla yhdessä samassa tiedostossa niin, että koodi tulostaisi rivi kerrallaan kokonaisen html-sivun, jossa on halutun tuotteen tiedot aseteltuna oikein HTML-lauseiden sisään. Tähän käytettiin PHP:n ”echo”-toimintoa. Jos siis valinnaisparametrille annetaan arvo ”b”, niin tällöin näkyviin tulostuu koko tuotetietosivu, eli tuotteen kuva, nimi, kustantaja, julkaisuvuosi sekä teoksen esittelyteksti, mikäli tällainen on saatavilla. Jos tuotteelta puuttuu yksi tai useampi edellisistä tiedoista, lukee näiden kohdalla tällöin ”ei saatavilla”.

Kyselyrajapinnan ohjelmakoodi muuttui luonnollisesti useampaan otteeseen projektin aikana, eikä läheskään aina niin selkeällä logiikalla kuin voisi luulla. Vaikka olinkin itse nimellisesti vastuussa kyselyrajapinnasta, oli jokaisella tiimimme jäsenellä pääsy kehityspalvelimelle, jossa tiedosto sijaitsi. Kaukaa viisaana pidin samanaikaisesti varmuuskopioita kaikista projektiin

liittyvistä tiedostoista omalla koneellani, ja yhden kerran jouduinkin palauttamaan varmuuskopion kehityspalvelimelle, kun kävi ilmi, ettei tämä jostain syystä toiminutkaan. Syynä tähän oli luonnollisesti se, että kyselyrajapintaan oli tehty tietämättäni muutoksia.

6.4 Tuotantoprosessin tuki

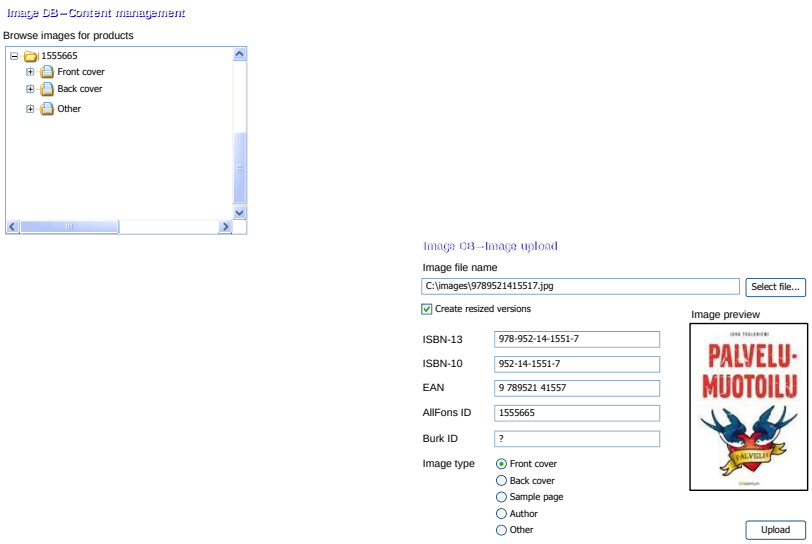
Projektin myöhemmässä vaiheessa yhtiön oman kuvatuotannon käyttöön haluttiin työkalu, jolla kuvapankin tuotteisiin voisi tehdä muutoksia tarvitsematta osata ohjelmoinnin perusteita. Käyttöliittymä on pääpiirteittäin mainittu jo aiemmin, ja kuviosta 3 on mahdollista hahmottaa paremmin kuvapankin etenkin tähän työkaluun liittyviä käyttöoikeustasoja. Käyttääkseen työkalun eri toimintoja on käyttäjän kirjauduttava ensin sisään. Käyttöoikeuksien hallinta oli tarkoitus toteuttaa tässä tapauksessa niin, että eri käyttäjätyypit saavat aina näkyviin eri määrän valittavia toimintoja. Käyttöliittymästä käytettiin virallisesti nimitystä tuotantoprosessin tuki.

Kuvio 7 havainnollistaa alkuperäistä suunnitelmaa kuvapankin graafisen käyttöliittymän ulkonäöstä. Varsinainen käyttöliittymä poikkesi kuitenkin jonkin verran ulkonäöltään alkuperäissuunnitelmasta. Valikkojen ulkoasu toteutettiin käyttämällä HTML-lauseiden lisäksi javascriptin valmiita kirjastoja. Lomake sisälsi esimerkiksi enemmän täytettäviä kenttiä tuotetietojen käsittelyä varten. Käyttöliittymällä on mahdollista lisätä, poistaa sekä tehdä muutoksia jo olemassa oleviin tuotteisiin sekä näiden medioihin. Näiden toteuttamiseksi oli ensin luotava hakutoiminto, joka ottaa HTML-lomakkeesta ensin vastaan yhden tai useamman tuotteeseen liittyvän tiedon, eli hakuehdon, joita on valittavissa kaikkiaan seitsemäntoista. Tämän jälkeen ohjelmakoodi luo annetuista hakuehdoista SQL-kyselylauseen samalla periaatteella kuin kyselyrajapinnassa.

Tuotantoprosessin tuen toimintaperiaate erosi kuitenkin kyselyrajapinnasta siinä, ettei jälkimmäisessä tarvinnut olla sisään kirjautumistoimintoa, sessionhallintaa eikä varsinaista hakutoimintoa. Tuotantoprosessin tuessa oli kyse useamman eri parametrin samanaikaisesta käsittelystä. Koodin oli kyettävä ottamaan lomakkeeseen annetussa syötteessä huomioon mahdolliset vajavaiset tai tyhjiksi jätetyt kentät. Ensin ohjelmisto käy läpi ja validoi kaiken annetun syötteen, eli siistii merkkijonoista mahdolliset kielletyt merkit sekä tyhjät välilyönnit. Tämän jälkeen siistityt merkkijonot luetaan kukin omaan muuttujaansa, jotka voidaan edelleen katenoida, eli yhdistää merkkijonoiksi, ja käyttää näitä hakuehtoina SQL-kyselylauseessa. Virheellisiä hakuehtoja syötettäessä ohjelma tulostaa ilmoituksen ”tuotetta ei ole”. Mikäli lomakkeen yrittää lähettää kaikki kentät tyhjinä, tulostaa ohjelmisto näkyviin virheilmoituksen ” et antanut mitään tuotetietoja”.

Mikäli annetut ehdot täsmäävät jonkin olemassa olevan tuotteen tietoihin, tulostetaan tuotteen kaikki tiedot näkyviin. Jos taas useampi eri tuote täyttää annetut ehdot, tulostetaan kaikki löytyneet tuotteet listanäkymään. Listalta on edelleen mahdollista valita haluttu tuote muokattavaksi tuotteen kuvassa sijaitsevan linkin avulla. Käyttöliittymän kehityshetkellä listanäkymä tulosti kerrallaan enimmillään kymmenen tuotetta sivua kohden, mutta tuotemäärän ylärajaa on mahdollista nostaa haluttaessa.

Muokkaussivulle siirryttäessä käyttöliittymä tulostaa kuvion 7 mukaisen lomakkeen, johon on valmiiksi täytettynä kaikki valitun tuotteen olemassa olevat tiedot. Kuvio 7 havainnollistaa alkuperäistä suunnitelmaa käyttöliittymän ulkonäöstä. Tuotantoprosessin tuen lopullisessa versiossa tuotetietolomake sisältää kentät kaikkiin mahdollisiin tietoihin joita yksittäinen tuote voi sisältää. Tuotetietojen lisäksi lomake sisältää tuotteen kuvan mikäli saatavilla, sekä toiminnallisuuden jolla voi tarvittaessa lisätä uuden kuvan. Käyttäjä voi tässä kohti valita joko muuttavansa ja edelleen päivittävänsä tuotteen tietoja, tai vaihtoehtoisesti poistaa tuotteen. Käyttäjä voi vaihtoehtoisesti poistua muokkaussivulta ja jättää tuotteen tiedot alkuperäiseen tilaansa.



Kuvio 7: Graafinen käyttöliittymä

Tuotantoprosessin tuki sisälsi toiminnallisuuksia, joilla on mahdollista hallinnoida asiakkaiden ja käyttäjien tilejä sekä näihin liittyviä käyttöoikeuksia. Asiakastilien hallinnointiin tarkoitettu työkalu avaa ensin ponnahdusvalikossa näkymän, josta käyttäjä voi valita joko vaihtoehdon ”lisää uusi”, ”asiakastilit” tai ”oikeudet”. Ensimmäinen vaihtoehto tulostaa lomakkeen asia-

kastilin luomista varten. Lomake sisältää kentät ”nimi”, ”salasana”, ”BTJ-tunnus”, ”salt” ja ”accno”, jossa ”accno” viittaa asiakasnumeroon. Seuraava vaihtoehto taas listaa kaikki olemassa olevat asiakastunnukset. Tästä näkymästä on mahdollista valita asiakastilejä poistettavaksi. Vaihtoehto ”oikeudet” luo taulukkonäkymän kaikkien asiakastilien käyttöoikeuksista. Käyttäjä valitsee ensin listalta asiakkaan, jonka tilille haluaa tehdä muutoksia. Sivulle tulostuu näkyviin myös taulukko johon on listattu kaikki mahdolliset tuotekategoriat, sekä näiden vieressä valintaruudut, joihin on merkitty rasti niiden kategorioiden kohdalle, joihin asiakkaalla sillä hetkellä on oikeudet. Muuttaakseen oikeuksia käyttäjän tarvitsee yksinkertaisesti lisätä tai poistaa ruksi jokaisen halutun tuotekategorian kohdalle, ja valita näkymän alareunasta vaihtoehto ”päivitä oikeudet”. Asiakkaan oikeuksien listaaminen oli toteutettava ohjelmoimalla useampi sisäkkäinen silmukka eli toiminto, joka suorittaa sille annetun tehtävän useita kertoja peräkkäin.

Käyttäjätilien hallinta toteutettiin hyvin pitkälti samalla tavalla kuin asiakastilien hallinta, paitsi ettei käyttäjätileille tarvinnut toteuttaa samanlaista tuotteiden käyttöoikeuksiin liittyvää muokkausmahdollisuutta. Tuotantoprosessin tuen päävalikosta on mahdollista valita käyttäjät–kohdasta joko vaihtoehto ”lisää uusi” tai ”käyttäjätilit”. Näistä molemmat pystyttiin toteuttamaan samalla periaatteella kuin asiakastilien vastaavat toiminnallisuudet. Ainoa huomionarvoinen seikka käyttäjätilien listaamisessa oli se, ettei ohjelmakoodi tulostaisi listalle mukaan sisään kirjautunutta käyttäjätunnusta. Käyttäjätilit sisälsivät samat tiedot kuin asiakastilit, lukuun ottamatta kenttää ”BTJ-tunnus”. Käyttäjätilien lisääminen tulostaa luonnollisesti täytettävän lomakkeen, ja tilien poistaminen onnistuu valitsemalla yksi tai useampi käyttäjätili listanäkymältä.

Tuotantoprosessin tukeen oli tarkoitus kehittää vielä joitakin lisäominaisuuksia. Yhtenä näistä oli toiminto jolla ohjelma ottaa vastaan CSV-tiedoston, lukee tämän ja tekee tietokantaan päivityksiä tiedoston sisällön mukaan. Salt–ominaisuudelle oli lisäksi tarkoitus luoda oma työkalunsa käyttäjätilien ja asiakastilien kohdille. Kaikkia näitä ominaisuuksia ei kuitenkaan ehditty toteuttaa projektin määräajan umpeuduttua.

7 Johtopäätökset

Vaikka nykyaikaisessa ohjelmistokehityksessä ns. ketterän menetelmän (agile software development) käyttö voikin olla jo lähes itsestäänselvyys, on sitä ilman mahdollista tulla toimeen. Pääasiassa vanhemman vesiputousmallin käytöstä kuitenkin luovuttiin ohjelmistokehityksessä 1990-luvun lopulla. Toisaalta kaikki ketterän mallin toimintaperiaatteista eivät vastaavasti ole sovellettavissa sellaisinaan ohjelmistokehitykseen.

Keskeistä onkin mielestäni erottaa ensin toisistaan käsitteet tuotantoprosessi (production process) ja kehitysprosessi (developement process), sekä tämän jälkeen sisäistää näiden väli-

nen yhteys. Poppendieckiä mukaillen voidaan ajatella että kehitysprosessissa luodaan ensin resepti tai periaate, jota sitten noudatetaan tuotantoprosessissa. Tässä tapauksessa kyse oli kuitenkin ehkä enemmänkin toiminnallisuudesta (kyselyrajapinta), tai tuotannon työkalusta ja apuvälineestä (tuotantoprosessin tuki). Periaate kuvatuotannosta sekä eri medioiden ja tuotetietojen syöttämisestä hallinnointiin oli olemassa jo ennen projektin aloittamista.

Karkeasti voisi sanoa, että vanhempi vesiputousmalli sopii paremmin tuotantoprosessiin, ja ketterä menetelmä taas paremmin kehittämiseen. Todellisuudessa vesiputousmallia käytetään kuitenkin yhä harvemmin. Tuotantoprosesseissakin ollaan aikoja sitten siirrytty ketterämpiin menetelmiin, joista yhtenä esimerkkinä aiemmin mainittu lean.

Kuvapankkiprojektissa CI-menetelmän käyttö osoittautui toimivaksi ratkaisuksi alusta alkaen. Samanlaista toimivuutta tuskin olisi voinut saada aikaan, mikäli ohjelmistoon ei olisi ollut mahdollista tehdä nopeita muutoksia vielä tuotantoon siirtämisen jälkeen. Vastaavasti muutoksia ja lisäominaisuuksia tuskin olisi ollut mahdollista saada aikaan yhtä tehokkaasti ilman jatkuvaa yhteyttä asiakkaaseen.

Etenkin erinäisten viivästymisien ja hankaluuksien voisi kuvitella vaikuttavan paljon näkyvämmin tuotantoprosessissa. Mikäli projektin johto ei kykene joko ajoissa puuttumaan ongelmiin, voivat tämän seuraukset olla tuhoisia.

8 Oman oppimisen arviointi

Kuvapankkiprojektiin osallistuminen antoi minulle ennen kaikkea tilaisuuden kokeilla oikeaa työntekoa sovelluskehityksen parissa. Viimeistään harjoittelujakson aikana tuli itsellenikin selväksi se tosiasia, ettei edes IT-alalla projekteja voida saattaa valmiiksi pelkästään yhden henkilön voimin. Vaikka kyselyrajapinta olikin melkein kokonaan minun vastuullani, jouduin jatkuvasti kyselemään muilta tiimin jäseniltä sekä esimieheltäni neuvoja eri toimintoihin ja periaatteisiin liittyen. Sama koski etenkin tuotantoprosessin tuki–toiminnon kehittämistä. Lisäksi pidimme säännöllisin väliajoin palavereita, joissa seurattiin projektin edistymistä ja esiteltiin aikaansaannoksia.

Tekemäni työtehtävät olivat mielestäni varsin vaihtelevia odotuksiini nähden. Kuvapankkiprojektin ohella jouduin esimerkiksi neuvomaan muita yrityksen työntekijöitä joissakin perusasioissa, kuten FTP-palvelimen käytössä tai langattoman verkon käyttöönottamisessa. Toki myös kyselyrajapinta oli varsin tärkeä projekti ja sen saattaminen valmiiksi asiakkaalle vähintäänkin vastuullinen tehtävä. Asiakas oli tässä tapauksessa vielä maksanut järjestelmästä etukäteen, ja odotti saavansa rajapinnan käyttöön mahdollisimman pian.

Varsinainen ohjelmointityö oli mielestäni ainakin riittävän haastavaa, ja harvoin ainakaan yksitoikkoista. Satunnaiset onnistumiset motivoivat työskentelyä varsin tehokkaasti, mutta vastaavasti monimutkaisempien ominaisuuksien toteuttaminen saattoi olla pahimmillaan todella hermoja raastavaa. Varsinkin kuvatuotannon graafisen ylläpitotyökalun saattaminen toimivaksi aiheutti useampaan otteeseen hankaluuksia. Yksinkertaiselta kuulostava idea voi usein olla ohjelmointityössä monimutkaisempi toteuttaa, kuin mitä voisi odottaa. Tällöin siihen uppoaa helposti tavallista enemmän aikaa.

Tietenkään tämänlainen työskentely ei välttämättä olisi mitään verrattuna niin sanottujen oikeiden IT-firmojen olosuhteisiin, joissa projektin deadlinen lähestyminen tarkoittaisi, että esimies hengittää jatkuvasti työntekijän niskaan. Vaikka projekti sitten lopulta saataisiinkin valmiiksi, ei etenkään puutteellisilla resursseilla toteutetun hankkeen toimivuudesta ole kunnollista varmuutta. Ei ole harvinaista, että työntekijä x osallistuu useampaan projektiin samanaikaisesti. Moni yritys saattaakin tänä päivänä omien resurssien lisäksi turvautua työtehtävien ulkoistamiseen halvemman työvoiman maihin, etenkin IT-alalla.

BTJ Finlandilla työilmapiiriä voisi sen sijaan luonnehtia enimmäkseen varsin rauhalliseksi. Tietysti oli sellaisiakin tilanteita, jolloin keskittyminen omaan työtehtävään oli hieman vaikeampaa. Monesti muiden osastojen työntekijät saattoivat esimerkiksi tulla kysymään apua johonkin tietotekniseen ongelmaan muilta tiimin jäseniltä.

Koin lopulta saaneeni paitsi aikaisempaa paremman käsityksen IT-alan työskentelytavoista, niin kykeneväni tehokkaampaan ajankäyttöön ja kokonaisuuksien hallintaan. Katson tottakai kokemuksen kartuttamisen ohjelmointityöskentelystä pelkästään positiiviseksi, joskaan en voi pitää itseäni vielä kovin taitavana koodaajana saati ohjelmistokehittäjänä.

Lähteet

Kirjalliset lähteet

Haikala, I & Märijärvi, J. 2004. Ohjelmistotuotanto. Helsinki : Talentum.

Heinisuo, R. 2007. PHP ja MYSQL: tietokantapohjaiset verkkopalvelut. Helsinki : Talentum.

Imai, M. 1986. Kaizen: The Key To Japan's Competitive Success. New York, itd : McGraw-Hill.

Lehtimäki, T. 2006. Ohjelmistoprojektit käytännössä : ohjeita ohjelmistoprojektien johtamiseen. Helsinki : Readme.fi.

Modig, N & Åhlström, P. 2013. Tätä on lean: ratkaisu tehokkuusparadoksiin. Suomentanut Maarit Tillman. Tukholma : Rheologica Publishing.

Neuendorf, S. 2004. Six Sigma for Project Managers (Project Management Essential Library). Vienna : Management Concepts.

Noronen, H-P. 2014. IMPROVING PERFORMANCE, QUALITY AND HAPPINESS OF SOFTWARE DEVELOPMENT TEAM : Agile and Lean approach. Jyväskylän ammattikorkeakoulu.

Poppendieck, T. & Poppendieck M. 2003. Lean Software Development: An Agile Toolkit. Addison-Wesley Professional.

Tuominen, K. 2010. Lean käytännössä. Helsinki : Readme.fi.

Väyrynen, V. 2009. Onnistunut testaus ja ketterät menetelmät. Haaga-helia ammattikorkeakoulu.

Sähköiset lähteet

Fermín, M. 2012. simpleimage.php. Viitattu 12.1.2016.
https://gist.github.com/miguelxt/908143

Kim, D. 2012. Continuously controlled integration for Agile development. Agility and Project Leadership Blog. Viitattu 3.1.2016.
http://www.projectmanagement.com/blog/Agility-and-Project-Leadership/6077/

Kivekäs, O. 2014. Avoin rajapinta. Viitattu 22.10.2015.
http://otsokivekas.fi/2014/06/avoin-rajapinta/

Krafcik, F. 1988. Triumph of the lean production system. Viitattu 25.3.2016.
http://www.lean.org/downloads/MITSloan.pdf

Vermeulen, D. & Collins, M. 2009. Lean Leanings. Viitattu 17.2.2016.
https://kim.kaizen.com/kimglobal/userfiles/Image/gl/article_leanmanagementauckland.pdf

Kuviot

Taulukot

Liitteet

Liite 1: Projektipäiväkirja

BTJ Armas -päiväkirja

19.06.2012:
Kuvapankkiprojektin kickstart, samalla ensimmäisen sprintin aloitus.

27.06.2012:
Ensimmäinen palaveri, speksausten purku.

28.06.2012:
Toinen palaveri, speksausten täydennystä, ohjeiden anto tietorakenteiden ja tietokantataulukoiden suunnittelua varten seuraavaksi kerraksi.

29.06.2012
Tietokantataulujen esittely palaverissa, taulujen tarkennuksen ja kenttien suunnittelun aloitus.

02.07.2012
Tietokantataulujen kenttien ensimmäisen version esittely, taulujen muokkaaminen erikseen omille välilehdilleen.

03.7.2012
Kenttien tarkempi speksaus ja käännös englanniksi ruotsalaisia varten.

04.7.2012
Tietokannan luominen palvelimelle MySQL work benchin avulla.

05.7.2012
Testidatan syöttäminen tietokantaan. Vikoja lauseiden käsittelyssä.

06.7.2012
Mysql-lauseiden vikojen korjausta.

24.7.2012
Ohjelmisto siirretty myös linux-palvelimelle.

26.7.2012

Kyselyrajapinta saatu toimimaan fopenin avulla niin että koodi palauttaa pelkän kuvan.

Liite 2: Kustannuslaskelma

Service for whole public library sector (huge for 10 million images)

Service	Description	Monthly Cost	Units	Total Cost/ Month	
V4-12	4 vCPU, 12 GB RAM	5 620,00 SEK	1	5 620,00 SEK	
TD-WIN 00-24 M-S	Technical Operation Windows (Capacity)	1 606,00 SEK	1	1 606,00 SEK	
High end Disc	Per allocated GB	6,50 SEK	1000	6 500,00 SEK	
Backup	Per used GB	3,10 SEK	1600	4 960,00 SEK	
IIS	IIS Webserver operation	133,00 SEK	1	133,00 SEK	
				18 819,00 SEK	**2 110,19 €**

Service for whole Helmet (tiny for 5 million images)

Service	Description	Monthly Cost	Units	Total Cost/ Month	
V1-4	1 vCPU, 4 GB RAM	2 290,00 SEK	1	2 290,00 SEK	
TD-WIN 00-24 M-S	Technical Operation Windows (Capacity)	1 606,00 SEK	1	1 606,00 SEK	
High end Disc	Per allocated GB	6,50 SEK	500	3 250,00 SEK	
Backup	Per used GB	3,10 SEK	800	2 480,00 SEK	
IIS	IIS Webserver operation	133,00 SEK	1	133,00 SEK	
				9 759,00 SEK	**1 094,28 €**

Service for whole Helmet (tiny for 10 million images)

Service	Description	Monthly Cost	Units	Total Cost/ Month	
V1-4	1 vCPU, 4 GB RAM	2 290,00 SEK	1	2 290,00 SEK	
TD-WIN 00-24 M-S	Technical Operation Windows (Capacity)	1 606,00 SEK	1	1 606,00 SEK	
High end Disc	Per allocated GB	6,50 SEK	1000	6 500,00 SEK	
Backup	Per used GB	3,10 SEK	1600	4 960,00 SEK	
IIS	IIS Webserver operation	133,00 SEK	1	133,00 SEK	
				15 489,00 SEK	**1 736,79 €**

Service for whole Helmet (small for 10 million images)

Service	Description	Monthly Cost	Units	Total Cost/ Month	
V2-8	2 vCPU, 8 GB RAM	4 013,00 SEK	1	4 013,00 SEK	
TD-WIN 00-24 M-S	Technical Operation Windows (Capacity)	1 606,00 SEK	1	1 606,00 SEK	
High end Disc	Per allocated GB	6,50 SEK	1000	6 500,00 SEK	
Backup	Per used GB	3,10 SEK	1600	4 960,00 SEK	
IIS	IIS Webserver operation	133,00 SEK	1	133,00 SEK	
				17 212,00 SEK	**1 929,99 €**